2

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

USER INTERFACE
TO AN
ICAI SYSTEM THAT TEACHES DISCRETE MATH

by

Roy Keith Calcote & Richard Anthony Howard

June 1990

Thesis Advisors: Hefner & Shing

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED

1b. RESTRICTIVE MARKINGS

2a SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for public release;
distribution is unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
Naval Postgraduate School

6b. OFFICE SYMBOL *(if applicable)*
MA and CS

7a. NAME OF MONITORING ORGANIZATION
Naval Postgraduate School

6c. ADDRESS *(City, State, and ZIP Code)*
Monterey, CA 93943-5000

7b. ADDRESS *(City, State, and ZIP Code)*

8a. NAME OF FUNDING/SPONSORING ORGANIZATION

8b. OFFICE SYMBOL *(if applicable)*

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS *(City, State, and ZIP Code)*

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
USER INTERFACE TO AN ICAI SYSTEM THAT TEACHES DISCRETE MATH

12. PERSONAL AUTHOR(S)
Calcote, Roy, K., and Howard, Richard, A.

13a. TYPE OF REPORT
Master's Thesis

13b. TIME COVERED
FROM ______ TO ______

14. DATE OF REPORT *(Year, Month, Day)*
June 1990

15. PAGE COUNT
368

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Artificial Intelligence, Intelligent Computer Aided Instruction,
ICAI Interface

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

The main thrust of this thesis is the design of a usable *Intelligent Computer Aided Instruction (ICAI)* user interface that does not use a natural language processor and runs on a personal computer. Discrete Mathematics is the knowledge domain for this project and the Discrete Math Tutor (DMT) is the name of the tutoring system. The DMT will allow the average student to benefit from a tutoring system now and not have to wait until the artificial intelligence researchers solve the natural language interface problem.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
Kim Hefner and Mantak Shing

22b. TELEPHONE *(Include Area Code)*
(408) 646-2361

22c. OFFICE SYMBOL
MA and CS

83 APR edition may be used until exhausted
All other editions are obsolete

6a. (continued) Computer Science and Mathematics Departments

USER INTERFACE TO AN
ICAI SYSTEM THAT TEACHES DISCRETE MATH

by
Keith Calcote
Lieutenant, United States Navy
B.S., Texas Tech, 1983
and
Richard Anthcny Howard
Captain, United States Army
B.S., United States Military Academy, 1982

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1990

Authors: ______________________
Roy K. Calcote

____
Richard A. Howard

Approved By: ______________________
Kim Hefner, Co-Thesis Advisor

____
Mantak Shing, Co-Thesis Advisor

____
Harold M. Fredricksen, Chairman,
Department of Mathematics

____
Fo Robert B. McGhee, Chairman,
Department of Computer Science

ABSTRACT

The main thrust of this thesis is the design of a usable *Intelligent Computer Aided Instruction (ICAI)* user interface that does not require a natural language processor and runs on a personal computer. Discrete Mathematics is the knowledge domain for this project and the Discrete Math Tutor (DMT) is the name of the tutoring system. The DMT will allow the average student to benefit from a tutoring system now and not have to wait until the artificial intelligence researchers solve the natural language interface problem.

TABLE OF CONTENTS

I. INTRODUCTION

A. TOO MUCH TO EXPECT

During the late 1970's, the United States Army experienced a phenomenon called ***Zero Defect Performance***. This phrase means that commanders accept no mistakes. Because of this policy, valuable Army personnel lost their careers and were forced to retire early. Since then, saner minds have prevailed and a new policy is in place. The Army calls the new policy the ***Band of Excellence***. The *Band of Excellence* refers to an imaginary zone of acceptable performance. Instead of 100% efficiency all the time, the Army considers any unit that stays within this imaginary performance zone as combat ready.

The *Zero Defect Performance* idea is analogous to the evolutionary process of ***Intelligent Computer Aided Instruction*** (*ICAI*) systems for the last 20 years. Many experts agree that since their inception, *ICAI* systems have not performed at 100% efficiency (Dede, 1986, pp. 329-353). R. Good describes three reasons why *ICAI* systems have not proliferated in the last decade:

1. There Exists No Common Database of How a Student Learns.
2. There Exists No Common Database of How a Student Learns.
3. There Exists No Efficient Natural Language Processor.
4. Machines can not learn. (Good, 1987, pp. 325-342)

A fourth reason for the lack of acceptance is that most of the existing systems like SOPHIE, STEAMER and GUIDON all run on large mainframes or specialized equipment (Weneger, 1986, pp. 12-45). Most students have no access to these types of machines. Thus, the state-of-the-art *ICAI* systems are locked away in research laboratories and away from the average student.

It is time for the evolution of *ICAI* systems to enter the era of the *Band of Excellence*. Instead of insisting that *ICAI* systems keep getting better, experts must decide on the level of acceptable performance. For example, since experts may not solve the natural language processor problem in the near future, perhaps it is not necessary to have an efficient natural language processor as part of the user interface to any *ICAI* system. Further, these acceptable programs must run on machines that are available to the common user.

The main thrust of this thesis is the design of a usable *ICAI* user interface that does not require a natural language processor and runs on a personal computer. Discrete Mathematics is the knowledge domain for this project and the *Discrete Math Tutor (DMT)* is the name of the tutoring system. The *DMT* will allow the average student to benefit from a tutoring system now and not have to wait until the artificial intelligence researchers solve some tough problems.

B. ICAI FEATURES

There are many ways to develop *ICAI* systems. However, most experts agree that every *ICAI* program must contain four basic parts: an *Expert Module*, a *Student Module*, a *Tutorial Module* and the *User Interface Module*. Figure 1 shows a generic representation of any *ICAI* system. (Duchastel, 1989, pp. 95-100)

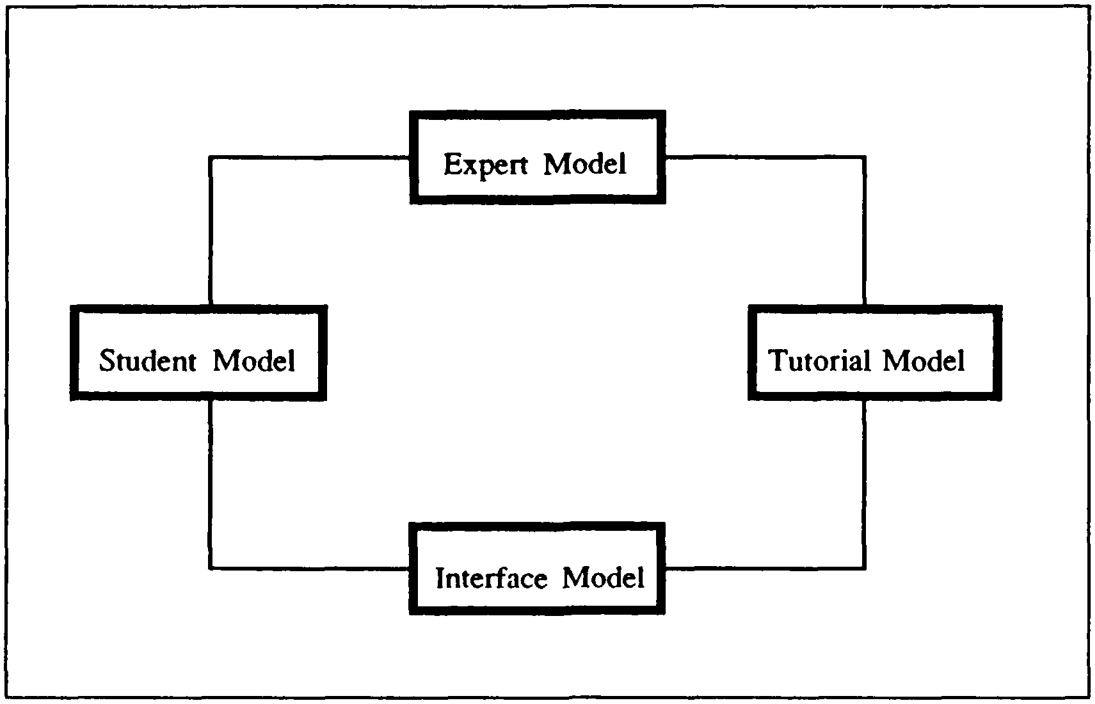


Figure 1 -General ICAI Model

The *Expert Module* is the problem solver. It contains all the information conceming how to solve problems in the subject domain and provides answers to student queries based on the subject domain. The *Student Module* contains all information about the student's performance level and reasoning strategies. It stores a list of a student's misconceptions and sub-optimal performance strategies that he uses to solve problems. It also stores any skills the student may already posses. The *Tutorial Module* contains instructional strategies to apply to each student based on the information contained in the *Student Module*. It makes inferences about the student's misconceptions and leaming needs, then selects the best instructional treatment for each student. The *User Interface Module* allows the student to interact with the other three modules. (Seidel, 1988, pp. 235-256)

C. THE INTELLIGENCE PART OF AN ICAI SYSTEM

The degree of Learner Control inherent in a *ICAI* system decides the intelligence of that system. Learner Control means that the student has control of the direction of the lesson path. In other words, the student is not dependent upon the software for each point in the lesson plan. LOGO is a good example of a system that provides complete Learner Control to the student (Jones, 1985, pp. 517-526). In LOGO, all learning is based on discovery. There exists no teaching strategy. The student merely tries what he thinks will work and makes corrections based upon the output of the program.

However, in intelligent tutoring systems, some teaching must occur. Thus, the tutoring system must dictate certain aspects of a lesson. When the tutoring system completely dictates the entire lesson and gives no control to the student, the system is simply called *Computer Aided Instruction (CAI).* Drill programs such as typing tutors and math skill programs are examples of *CAI* systems. (Duchastel, pp. 93-98)

An *ICAI* system requires a compromise between *CAI* and complete Learner Control. The compromise is called a *mixed-initiative environment* (Duchastel, pp. 93-98). The *mixed-initiative environment* provides some control to the student concerning how a lesson progresses; but, also, provides control to the tutor during key points in a lesson. Therefore, a tutoring system must implement a *mixed-initiative environment* in order for it to be classified as intelligent.

D. THE DISCRETE MATH TUTOR (DMT)

The *DMT* provides a simple *mixed-initiative environment.* The environment contains two parts. The first part presents a standard *CAI* lesson to the student. The *DMT* presents each lesson as pages on the screen. The student is allowed to page up, page down, and even view a particular page in a lesson.

The second part provides the *mixed-initiative environment.* From inside the lesson, the student may access the *DMT* user interface. The *DMT* user interface contains the following functionality:

1. Provides the user with the ability to print or save to disk any definition, algorithm or example in the *DMT*.
2. Provides the user with the ability to run any algorithm in the *DMT* on his own data.
3. Provides the user with other tools that aid the learning process.

These three functions allow the user to stop the lesson he is currently working on at any time and pursue topics of interest. For example, during a lesson a user may find a term he does not remember. All he has to do is access the *DMT* user interface and look up the definition of the unknown term. Further, during a problem solving session, a user can access the user interface's algorithm section and determine if his answer is correct. Or, the user can adjust the parameters of an example given in the lesson and view the resulting change. Finally, the user can access any tool provided by the lesson author that enhances the subject domain.

These three functions provide a simple, but effective, *mixed-initiative environment* that allows a user to learn discrete math. Further, the *DMT* runs on one of the IBM PC or compatible computers that are prodigious throughout the academic community.

The *DMT*'s *mixed-initiative environment* places the *ICAI* system into the *Band of Excellence*. Granted, the *DMT*'s representation of the *Expert Module*, the *Student Module* and the *Tutorial Module* is unsophisticated at this time. However, the *DMT* is available now to the average student on a computer system that is readily accessible.

II. HOW TO WRITE A LESSON

A. INTRODUCTION

The tutorial interface is designed so that easy use is achieved. Step-by-step instructions are provided for adding either lessons or question and answer sessions to the tutorial interface. An explanation of the use of the interface is provided in Chapter III: Users Manual. A basic knowledge of a word processor is required to take advantage of the interface so as to create a text type tutorial in any given subject. Producing graphic type tutorials or graphic drill sessions requires that the graphics lesson be independent of the interface. Programming experience in a computer language such as *C* or *Pascal* is necessary to create graphics lessons.

B. CREATING TEXT LESSONS

A lesson can be created with many common word processors. WordPerfect 5.0, WordStar 4.2, or MultiMate 3.30 are a sample of those which may be used. The word processor or text editor that is used must be able to create ASCII (American Standard Codes for Information Interchange) files.

In WordPerfect 5.0 text files are created by using the text in/out key (Ctrl-F5) (Kelly, 1988, p.498). MultiMate 3.30 requires that a document be converted to ASCII by using the advanced utilities menu, file conversion option. (Multimate International Corporation, 1984, p. A-3-45). The ASCII format was chosen so that a high degree of portability is assured, and that ease in creating a tutorial is achieved.

A single tutorial subject may be up to 100 pages long. Each page will be no longer than 19 lines. The first line, which is automatically created by the interface, will contain the page number. The remaining 18 lines may be used for the lesson text.

Each line may be up to 76 columns in width. The limitation for the number of lines and the column width is so that the page will fit into a window which is 19 rows by 80 columns. The window in which the pages are placed is created by the interface.

Each page in the lesson should be separated by a page break. The page break created by the word processor should be interpreted in the ASCII conversion process as the hexadecimal number 0C. The page break, 0C hexadecimal, is also known as a form feed (Hansen, 1989). If the lesson writer prefers to create a lesson with no page breaks, the lesson format program *txtmod.exe* will size the lessons to the correct length of 18 lines. If *txtmod.exe* is relied upon to build the pages, then it is possible that the material will not be presented in the manner in which the lesson-writer intends.

C. FORMATTING THE LESSON

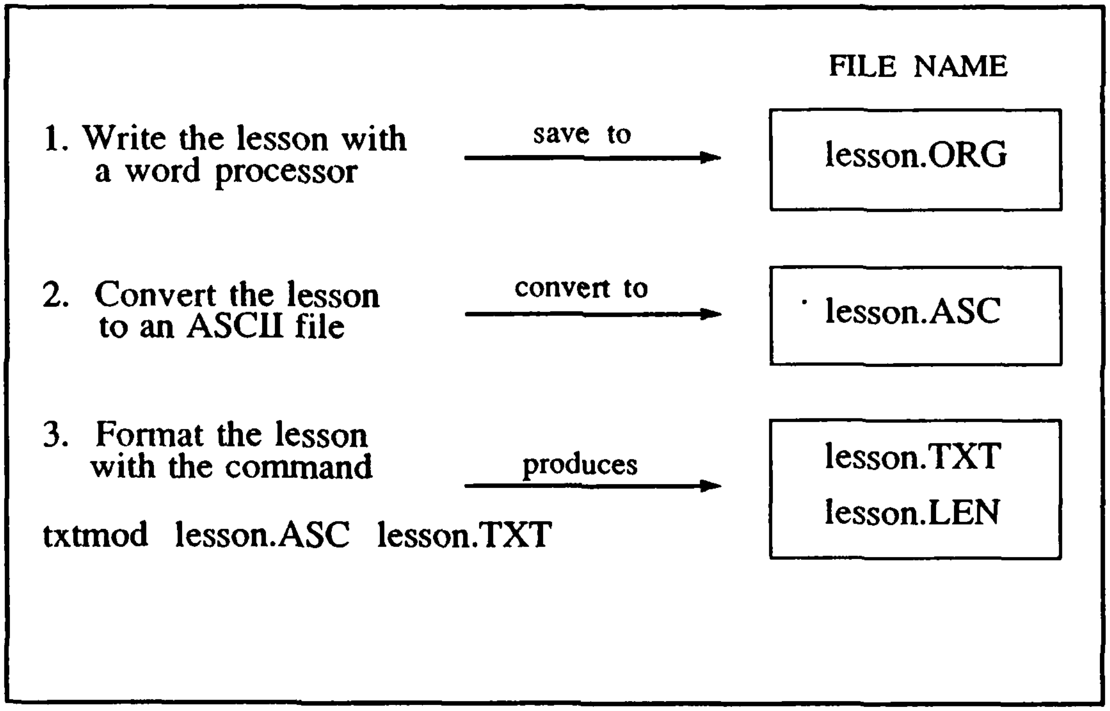
After a lesson is developed, the lesson format program *txtmod.exe* must be executed so that the interface and the lesson are properly aligned. For example, suppose that a lesson has been written with a word processor and given the file name *lesson.ORG*. Next *lesson.ORG* is converted to text format (an ASCII file) and assigned a new name: *lesson.ASC*. Now the lesson is formatted with the following command:

```
txtmod  lesson.ASC  lesson.TXT
```

lesson.ASC is the input file and *lesson.TXT* is the output file to the executable program txtmod. *lesson.TXT* is the file that will be used by the interface to present the lesson. This process is summarized in Figure 2.

The input file and output file must have different names. If the names were the same, the executable file *txtmod.exe* will try to overwrite the input file as it is being read and will produce unpredictable results. As this is the case, *txtmod.exe* will not allow the same name for the input and output files.

Other than the output file, another file called the *length file* is generated by *txtmod.exe*. The *length file* is created automatically and without effort on the part of the user. The *length file* consists of an array which contains the number of bytes between page breaks. It is the information which is stored in the *length file* that allows the interface to provide for next page, previous page, and individual page selection. The *length file* is given the same name as the output file but is given the extension *LEN* .


1. Write the lesson with a word processor
2. Convert the lesson to an ASCII file
3. Format the lesson with the command

```
txtmod  lesson.ASC  lesson.TXT
```

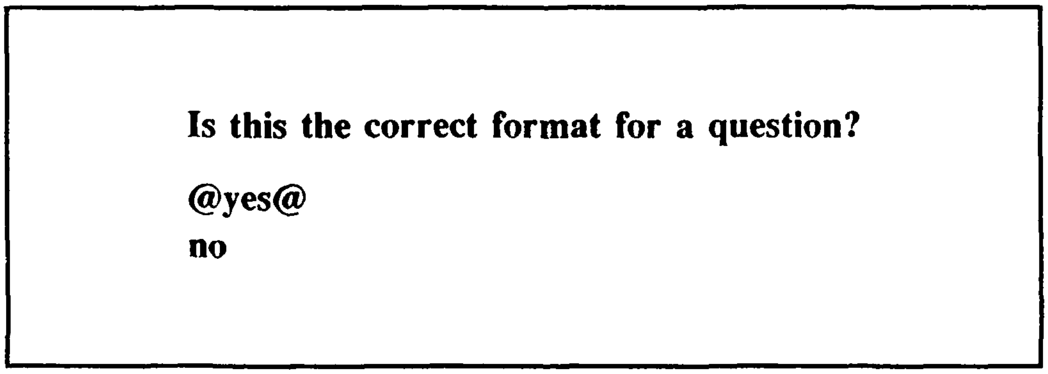
Figure 2 -Process of Formatting a Lesson

D. CREATING TEXT EXAMS

The interface has provisions for presenting text type question and answer exams. The exams may be of type true/false and/or multiple choice. An *exam bank* is a file which may consist of up to 100 questions. The student will be allowed to select any number up to the number of questions in the *exam bank.* The questions will be presented in random order. At the desire of the lesson-writer, explanations may be provided for the questions.

An *explanation bank* is a file which contains explanations for the corresponding questions. Providing an *explanation bank* is optional. But, if an *explanation bank* is given, an explanation page must be provided for each question in the corresponding question bank. This is required so that the questions and explanations are properly aligned. If it is preferable to provide explanations for some questions and not for others, a statement such as "Explanation not provided for this question" should be used for the appropriate page in the *explanation bank.*

Producing questions, answers, and explanations with the interface is similar to constructing lessons except that a specific format must be used to create a question. Questions and explanations may be at most 18 lines long. As shown in Figure 3, correct answers to questions must be bracketed with the @ (at sign) symbol.

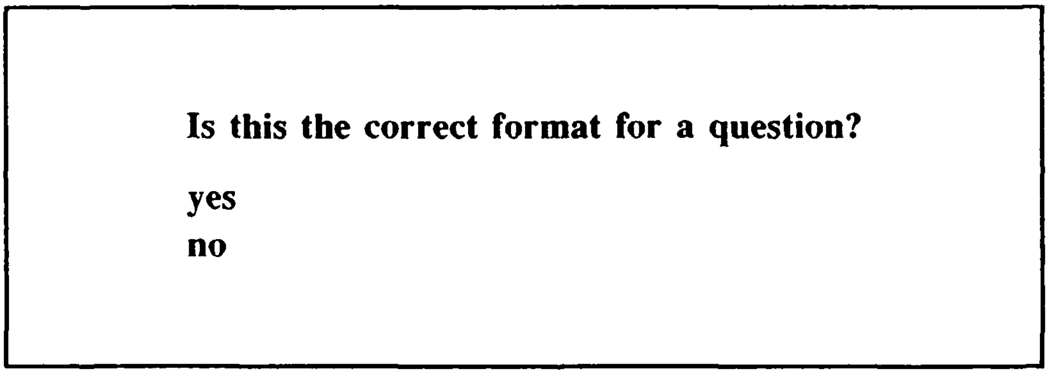


Is this the correct format for a question?

@yes@
no

Figure 3 -Question Format for Exam Bank

Notice that no blanks are allowed between the @ (at sign) and the correct answer. The interface, through use of the executable program *Q_&_A.exe* , will strip the @ brackets from the correct answer and present the correct answer so that it is aligned with the other answers. This is shown in Figure 4. The @ is required so that the correct answer is highlighted (reverse video) in response to an answer proposed by the user (see Figure 15). A word processor such as those mentioned above should be used to create ASCII files for both the *question bank* and the *explanation bank.* Then the lesson format program *txtmod.exe* must be executed on each of the ASCII files so that the interface and the exam are properly aligned. As with the lesson, *length files* are created by *txtmod.exe* for both the question and explanation files.



Is this the correct format for a question?

yes
no

Figure 4 -Question Presentation in the DMT

E. CREATING GRAPHICS LESSONS AND EXAMS

In the introduction, it was noted that a graphics lesson must be independent of the interface. That is, any graphically oriented lesson must be provided in an executable file. This is because there are no special provisions or restrictions in the interface for producing graphics. Also, because graphical lessons are independent of the interface, the lesson-writer may produce graphics lessons with a computer language of choice.

As illustrated in Figure 17, a graphics lesson is not displayed in the lesson window that is provided with the interface. This is because the windows and menus created by the interface are developed in the text mode instead of the graphics mode. Consequently, a program that produces graphical lessons or exams must first identify the graphics hardware installed on the computer. Next, the program must clear the text screen and initialize the graphics system. After initializing the graphics system, presentation of the graphically oriented lesson is possible. It should be noted that since a graphics lesson is not presented in the interface window, the aforementioned display restrictions of 19 rows by 76 columns do not apply.

After the lesson is presented and it is desired to return to the interface, the graphics program must clear the graphics device from the system. After clearing the graphics device, the text mode must be reinstalled. The interface will then reestablish the windows and menus and return the user to the lesson from which he came. Examples of graphic type lessons and exams are provided in Chapter III: Users Guide.

F. MEMORY CONSIDERATIONS

The DMT was designed to run on a *personal computer (PC)* with 640 kilobytes of RAM. Thus, 640K is an upward limit for how large the program can grow. The DMT currently uses 200 K of RAM while executing. The fact that only one of four modules of the DMT has been implemented makes the remaining 440K of RAM a critical commodity.

With this is mind, the DMT was designed with a special programming technique used in large programs called *layering*. It involves converting the major functions of a program into executable files. Instead of the main module of the interface calling individual *C Language* functions, the main module actually suspends operation of itself and calls other layered programs. Once the layered program finishes executing, control is returned to the DMT's main module. Thus, as long as the main module and any other layered program together do not exceed the 640K upper bound, the number of layered programs that can be added to the complete program is unlimited.

The DMT consists of a main program and a number of layered programs. The main module is called *dmt.exe* and is the actual interface to the program. When the program begins, *dmt.exe* always exists in RAM.

The key layered program in the tutor is called *lsn.exe* and is the executable file that displays lessons to the user. This layered program is critical because it will usually always co-exist in RAM with the *dmt.exe* interface program since displaying lessons is the main function to the tutor.

As shown in Figure 5, both *dmt.exe* and *lsn.exe* combine to use 200K of RAM. Therefore, only 440K of RAM is available for all other layered programs in the tutor. This seems to be sufficient since all the other layered programs that already exist inside the tutor are well below the 440K maximum.

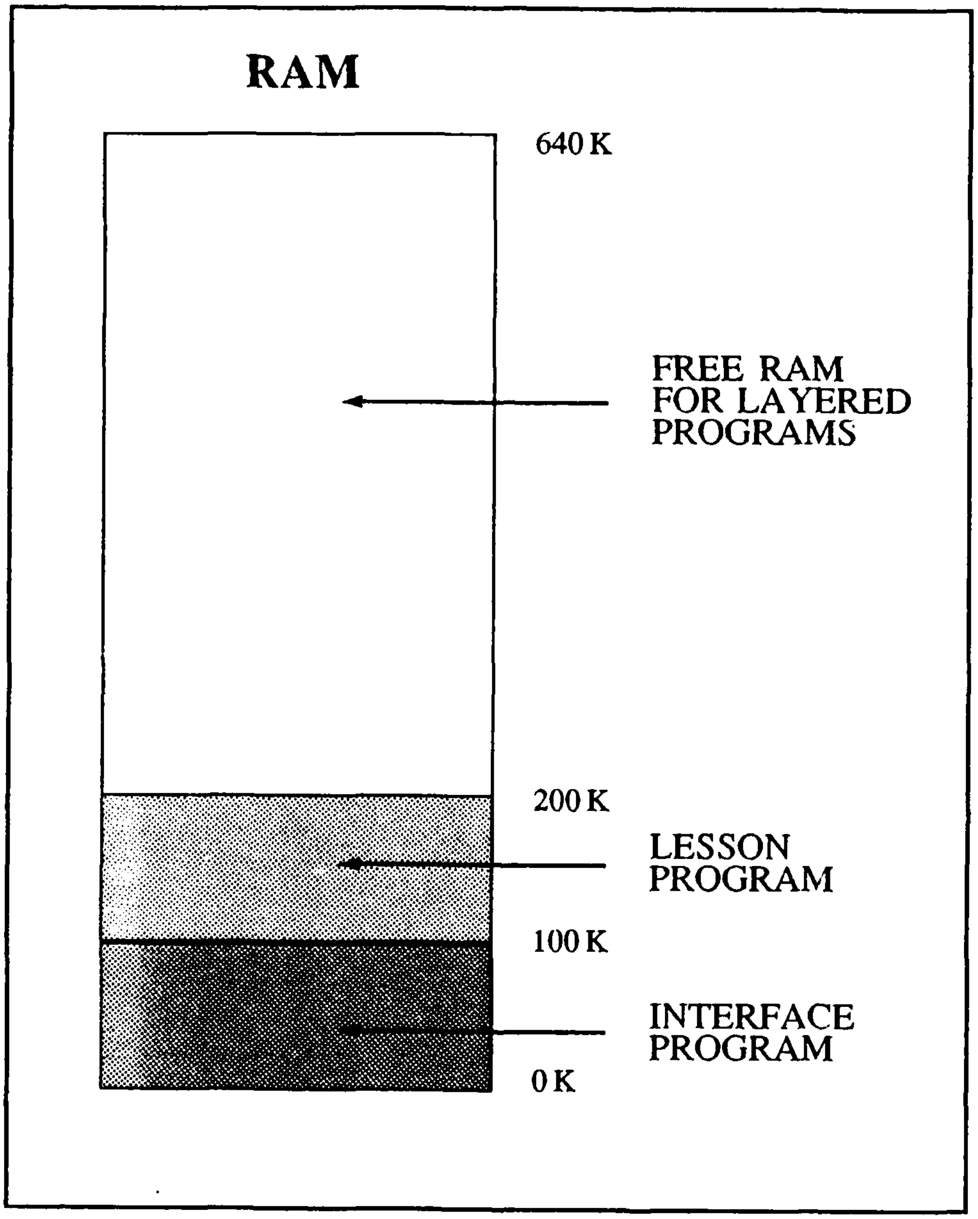


Figure 5 -DMT Memory Model

G. ADDING GRAPHICS TO LESSONS

The DMT utilizes Mike Smedley's windowing package called the C Extended Library (CXL) (Smedley, 1989). As mentioned previously, CXL does not support both graphical and textual modes simultaneously. If a picture is required to enhance a lesson, that picture is included as a tool inside the tutor interface. The lesson text must inform the student to locate the needed image inside the tool box for viewing. Thus, all pictures required in a lesson become small executable files, layered programs (see previous section), that are called from the "TOOLS" pull-down menu inside the DMT.

H. ADDING LESSONS, EXAMS AND TOOLS TO THE INTERFACE

Adding lessons, exams and tools to the interface is a two step process. First, the author must create these items. The previous sections in this chapter discuss this process. Second, the author must add each new item to the menuing system presented to the student.

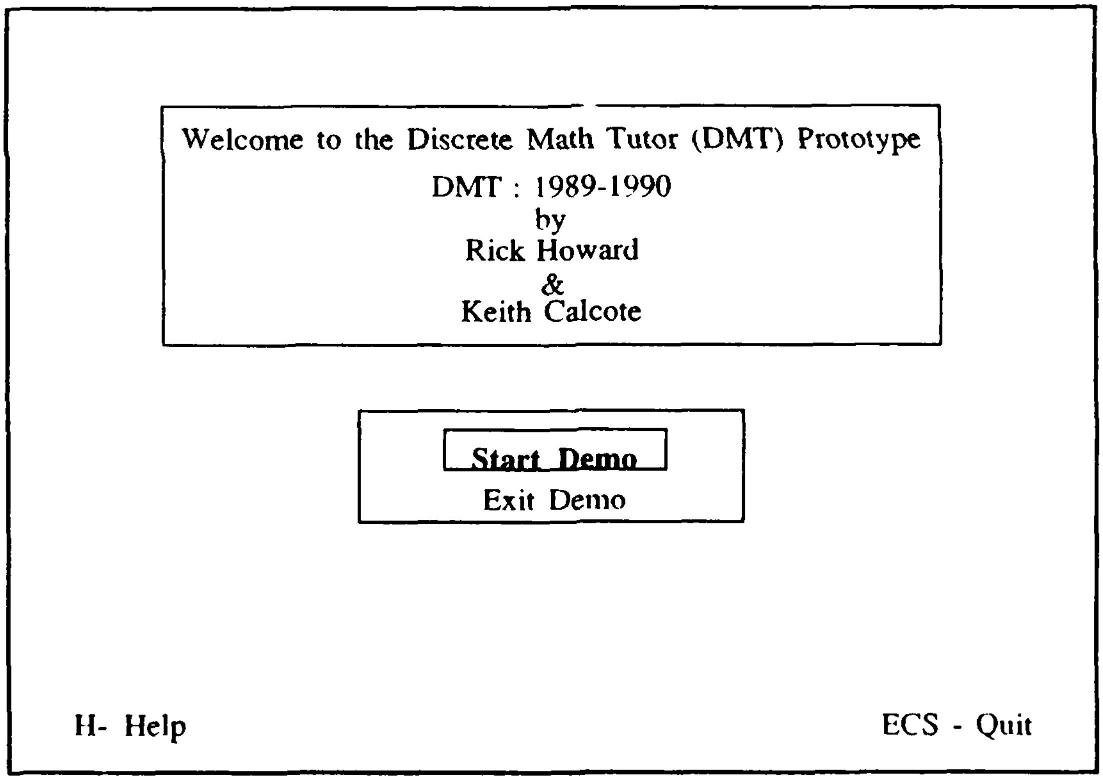
The CXL package provides an easy mechanism to provide pull-down menus to the user. The basic structure to each pull-down menu already exists in the DMT and is well documented in the CXL Documentation Book (Smedley, 1989). In general, however, the author calls a CXL function that defines the menu name and the name of the function that will execute once the menu item is chosen. This executing function uses the *spawnl* function provided by the Turbo C Library to suspend operation of the interface program and run some other executable file. This process is called *spawning* a program.

In the case of a new lesson, the executable file, *lsn.exe*, is called with the name of the new lesson's text file included as a command line argument. In the case of a new exam, the executable file, *exam.exe*, is called with the name of the new exam's text file included as a command line argument. Finally, for new tools, the tool's executable file is called with no command line argument. In all cases, control is returned to the DMT interface once the *spawned* program terminates.

III. USERS GUIDE

A. INTRODUCTION

The Discrete Math Tutor is started from the operating system command line prompt by typing the command, *DMT* , inside the directory that holds the *DMT* files (See Appendix C for details on installing the program on a hard drive). The first screen that appears is the introduction and is shown in Figure 6. From the introduction, the user has four options which may be selected by pressing designated keys known as *hot keys*.



Welcome to the Discrete Math Tutor (DMT) Prototype
DMT : 1989-1990
by
Rick Howard
&
Keith Calcote

Figure 6 -Introduction Screen

The first *hot key,* case sensitive *help,* is invoked by typing the letter H. Help is available throughout the tutor, and the following description is common to all help menus. If *help* is selected, an introduction to the available help is displayed. With the exception of the *Esc* key, the operation of all *hot keys* is suspended while *help* is active. The help screen describes the *hot keys* and explains other information pertinent to interface operations that are specific to the particular location in the tutor. If additional help is available, PgUp and/or PgDn is displayed at the bottom right hand corner of the screen. PgUp indicates that the previous page of *help* may be selected by typing the *page up* key. PgDn signifies that the next page of help is available by typing the *page down* key. Typing the *Esc* key will exit the *help* screen. Typing the *Esc* key again will exit the Tutor and return the user to the operating system.

The second *hot key, escape (Esc),* is available from the introduction screen to quit the Tutor. Selecting *Esc* from the introduction will return the user to the operating system. Also, the *escape* key is accessible throughout the tutor to back out of the menus.

The third hot key for the introduction is E. Typing E or selecting *Exit Demo* with the cursor and pressing enter will quit the Tutor and return the user to the operating system.

The last hot key for the introduction, S, is used to start the demonstration. The demonstration may also be started by selecting *Start Demo* with the cursor and pressing enter. When *Start Demo* is selected a blank opening screen is displayed. Figure 7 shows the tutorial opening screen.

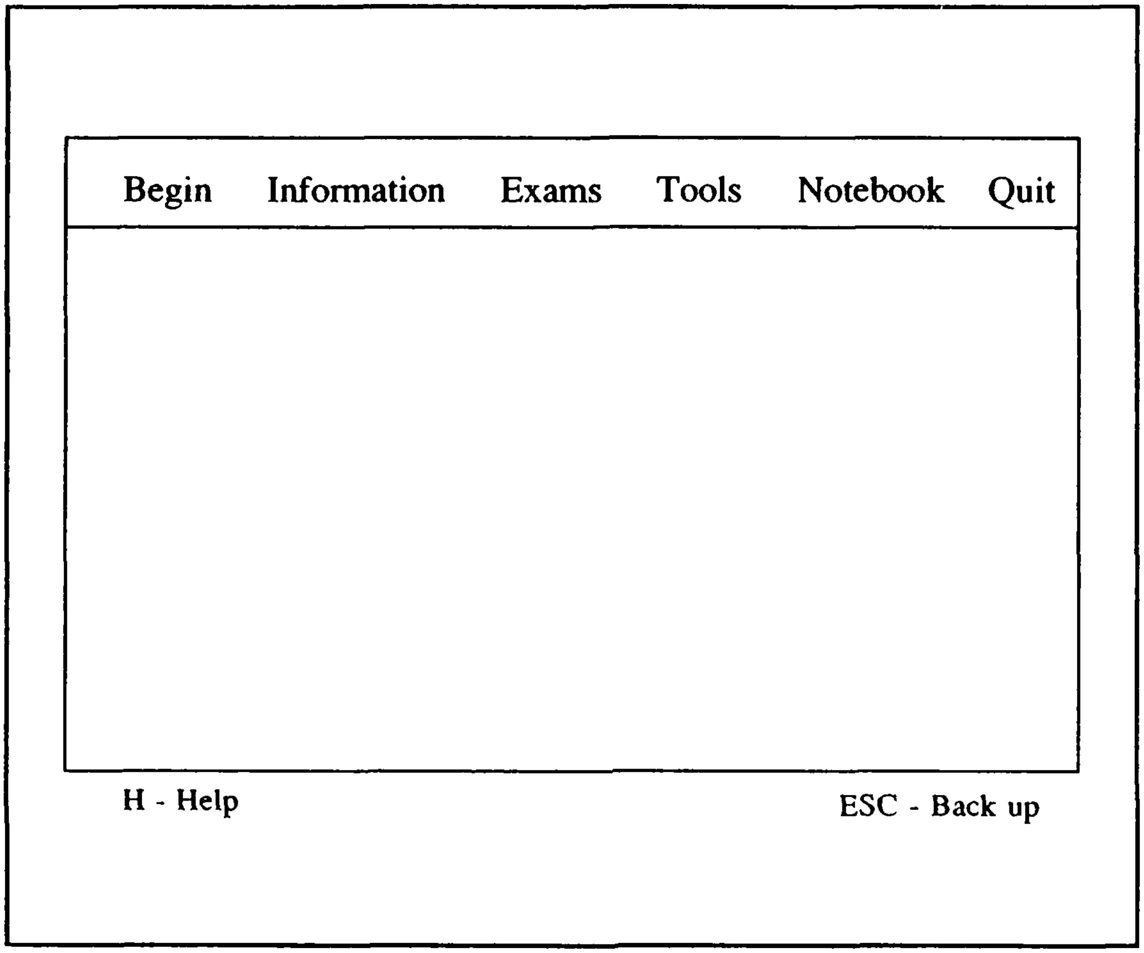


Figure 7 -Opening Screen

The menu b:. across the top of the opening screen is called the *main menu* and contains selectable options incorporated in the interface. The blank portion of the screen is where the lesson is displayed. The bottom portion of the screen includes additional information or directions available to the user.

B. BEGIN

The interface provides two ways to begin a lesson. A lesson may be started with the first page in the lesson, or it may be started with the last active page of previous session. To start the lesson, *Begin* is selected from the *main menu* by typing the *hot key, B*. Figure 8 displays the opening screen with the Begin menu selected.

From the *begin* menu the user has two options, *Start a Lesson* or *Return to Last Lesson*. Either of these two options are selected by cursor or by *hot keys* (S for *Start a Lesson* or R for *Return to Last Lesson*).

If *Start a Lesson* is chosen, a menu of available lessons is presented. This is shown in Figure 9. Once the available lessons are listed, the selection is made by moving the cursor to the desired lesson and pressing enter.

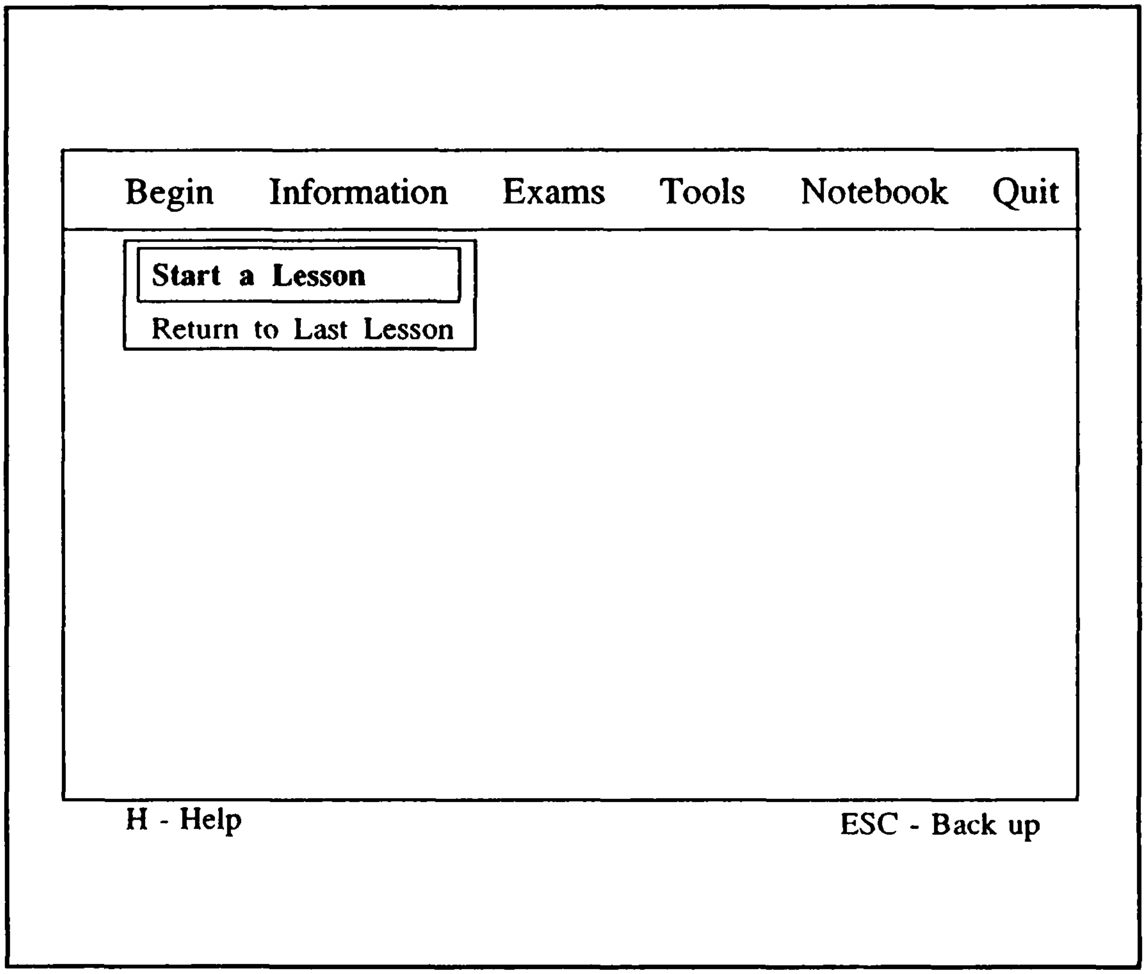


Figure 8 -Selection of the Begin Menu

Figure 10 displays the screen that is presented when *Return to Last Lesson* is selected from the *Begin* menu. The user is prompted to enter his or her social security number (ssn). The ssn is used to provide unique cataloging of multiple users. When the user enters his or her ssn, the interface locates and displays the last active page of the last lesson that corresponds to that ssn.

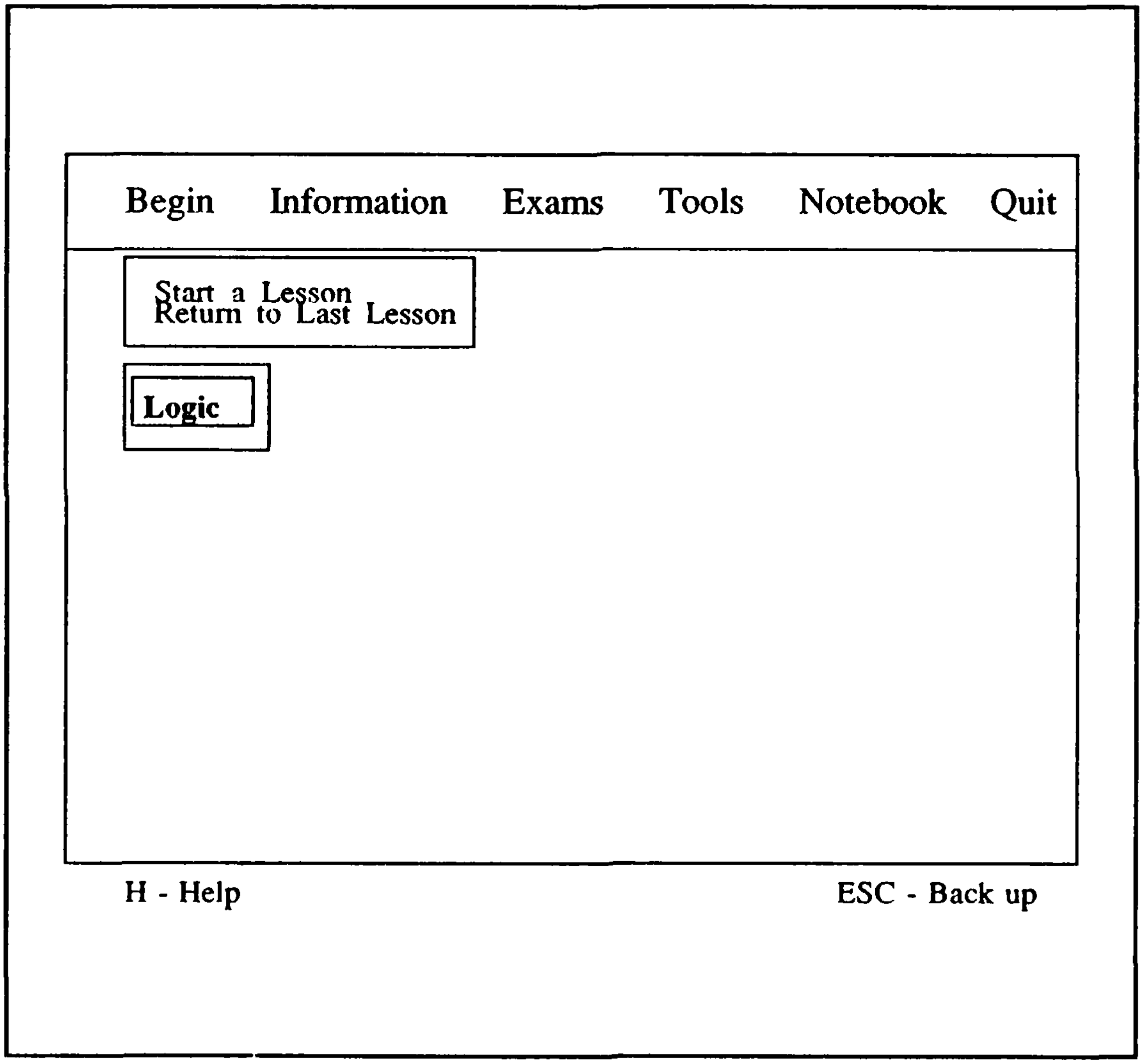
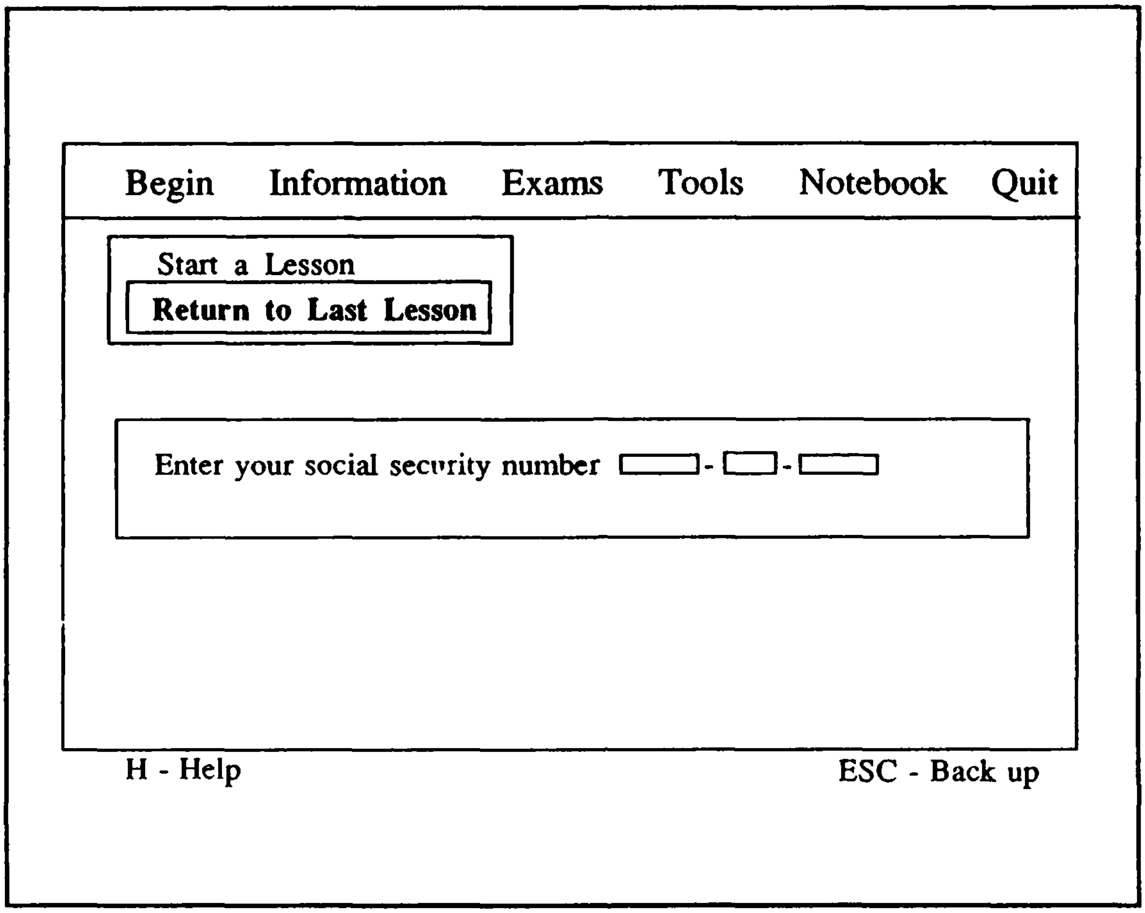


Figure 9 -Logic Lesson Selected from Menu

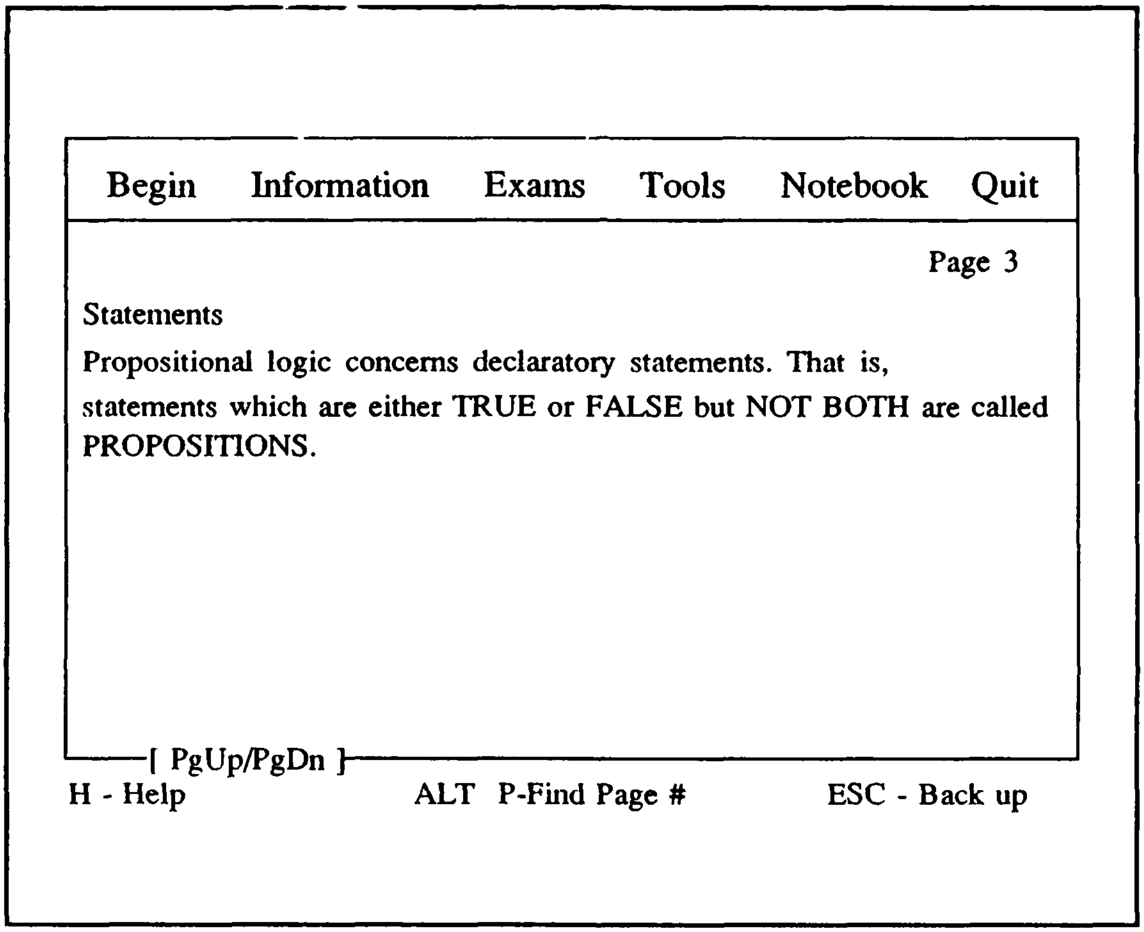


Enter your social security number □-□-□

Figure 10 - Social Security Number Required to Return to Last Lesson

C. MANEUVERING INSIDE A LESSON

A sample of a display of a lesson is provided in Figure 11. The page number is listed in the top right hand corner of the lesson. Pages may be selected by typing the two key combination *Alt P*, and entering the desired page number. If a page is selected that is out of bounds of the present lesson (e.g., page 60 is selected but the lesson is only 40 pages long), the lesson is started over at page one. The *page up* key may be used to select the previous page and the *page down* key may be used to select the next page. If the *page up* key is used when the lesson is on the first page, the lesson is wrapped to the last page. Similarly, if the *page down* key is used when the lesson is on the last page, the lesson is wrapped to the first page.



Statements

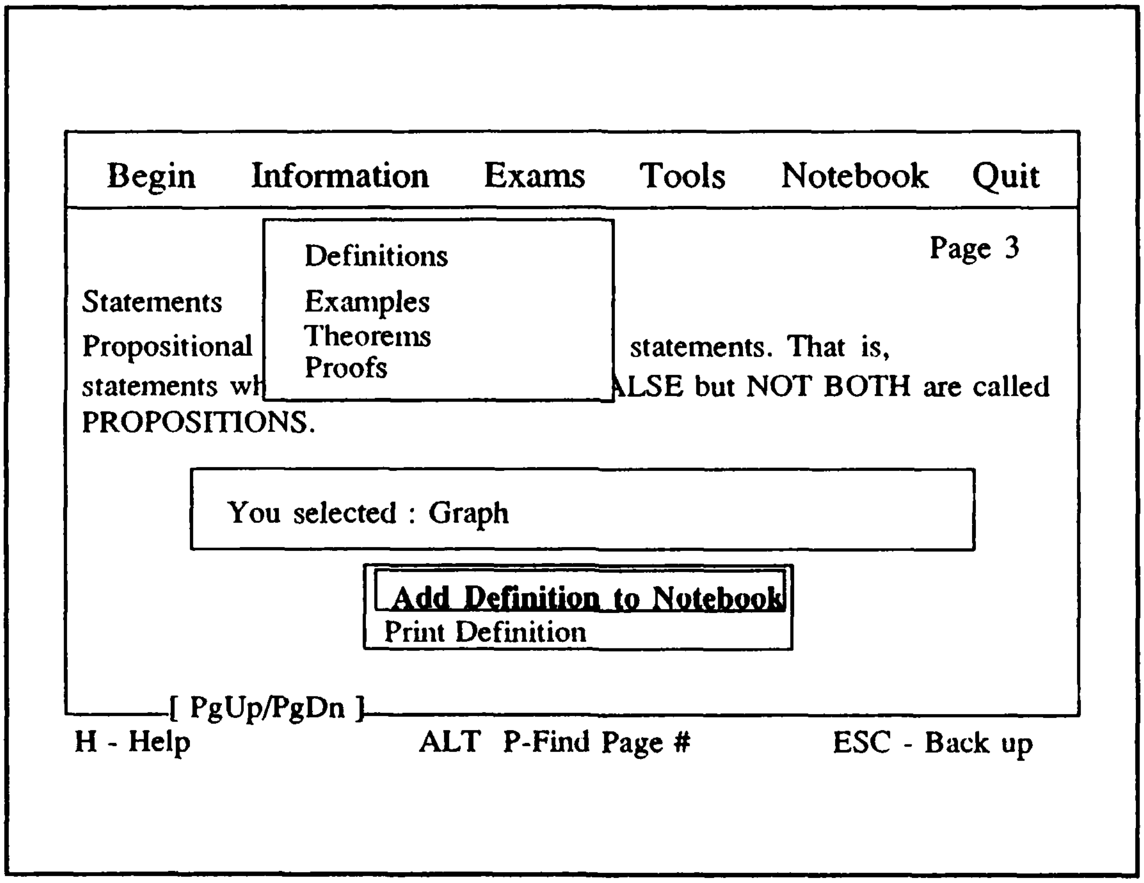
Propositional logic concerns declaratory statements. That is, statements which are either TRUE or FALSE but NOT BOTH are called PROPOSITIONS.

Figure 11 -Display of a Sample Lesson

D. INFORMATION

The information section was designed so that the student user has review material available for quick access. The selected material may be added to the user's notebook (described later in this section) or may be *directed to the printer* for hard copy. This feature allows the student to store and later retrieve material that he or she identifies as needing additional study.

Information is chosen from the *main menu* by typing the hot key I. Definitions, examples, theorems, and proofs are selectable from the *information* menu. Once the type of information is selected, a list of available items are displayed. The user makes a selection by moving the cursor to the desired item on the list and typing enter. As shown in Figure 12, the definition of a graph is chosen from the list of definitions while a lesson remains active in the background.

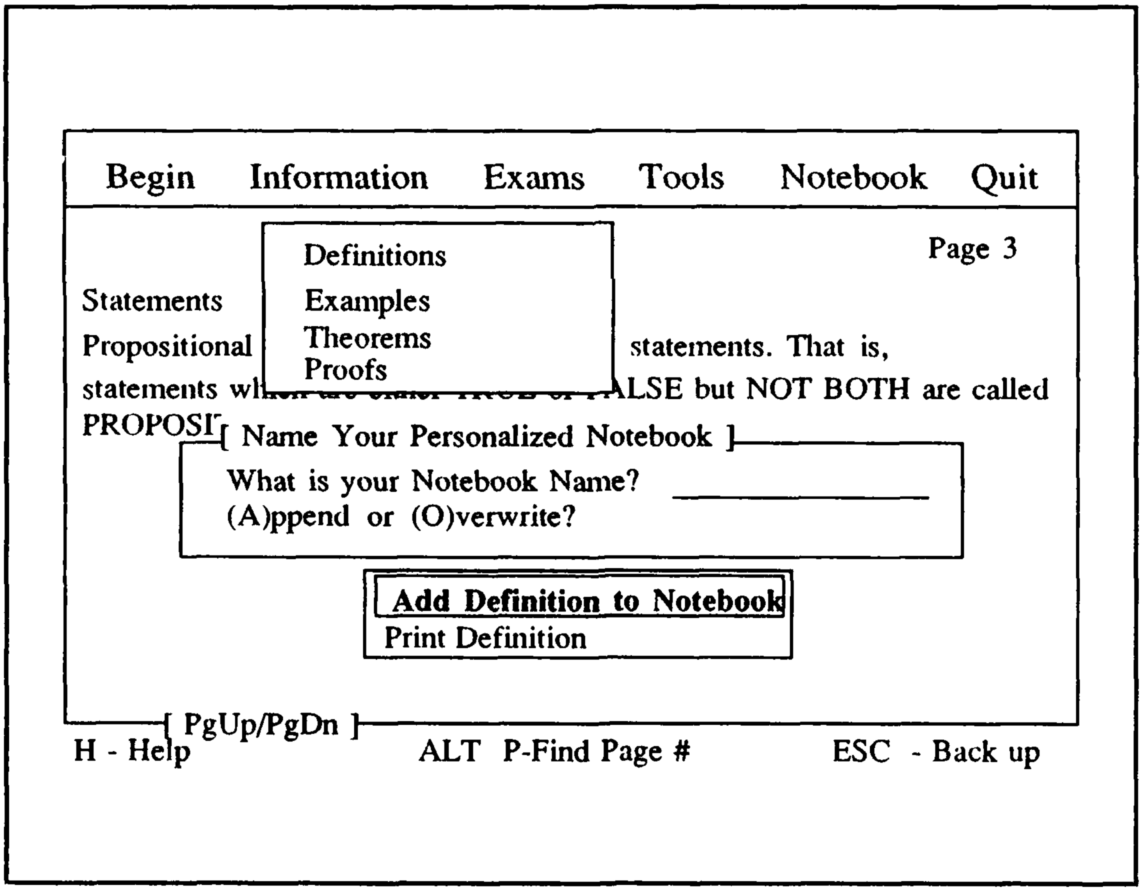


Statements
Propositional statements. That is,
statements wh LSE but NOT BOTH are called
PROPOSITIONS.

You selected : Graph

Figure 12 -The Definition of a Graph is Selected to Add to the Notebook

After the selection is made, the user is presented with two options. The item may be *added to the notebook* with the *hot key, N,* or *directed to the printer* with the *hot* *key, P*. Also, either of the options may be selected by the cursor. The user is cautioned to ensure that the printer is turned on prior to typing the *hot key, P*. If the notebook option is selected, the user is asked to provide the name of the notebook and asked whether the item should be appended to or overwrite the notebook. Again, the user is cautioned that if the overwrite option is chosen, all contents of the notebook are erased prior to writing the item to the notebook. Figure 13 displays the screen that results when *add to the notebook* is selected. After the task is completed, the user is returned to the page in the lesson from which he came.



Statements

Propositional statements. That is, statements wh ALSE but NOT BOTH are called PROPOSI

Name Your Personalized Notebook

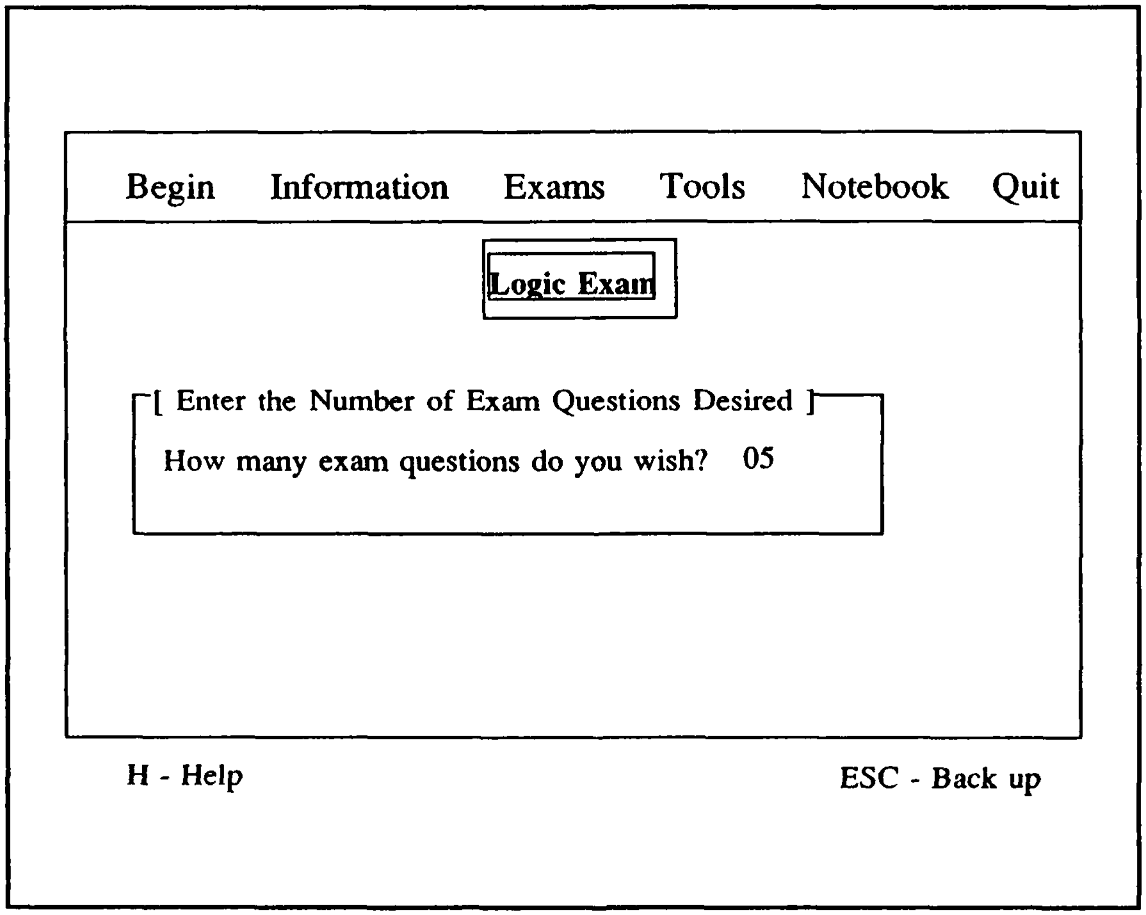
What is your Notebook Name? ____________

(A)ppend or (O)verwrite?

Figure 13 -Notebook Name is Requested

E. EXAMS

The *exams* section is provided to allow the user to test his or her knowledge of a particular area. *Exams* is selected from the *main menu* by typing the *hot key, E*. Subsequently, a list of exams are presented. A particular exam may be chosen by moving the cursor to the name of the exam in the list and pressing the enter key. Selection of an exam is shown in Figure 14.



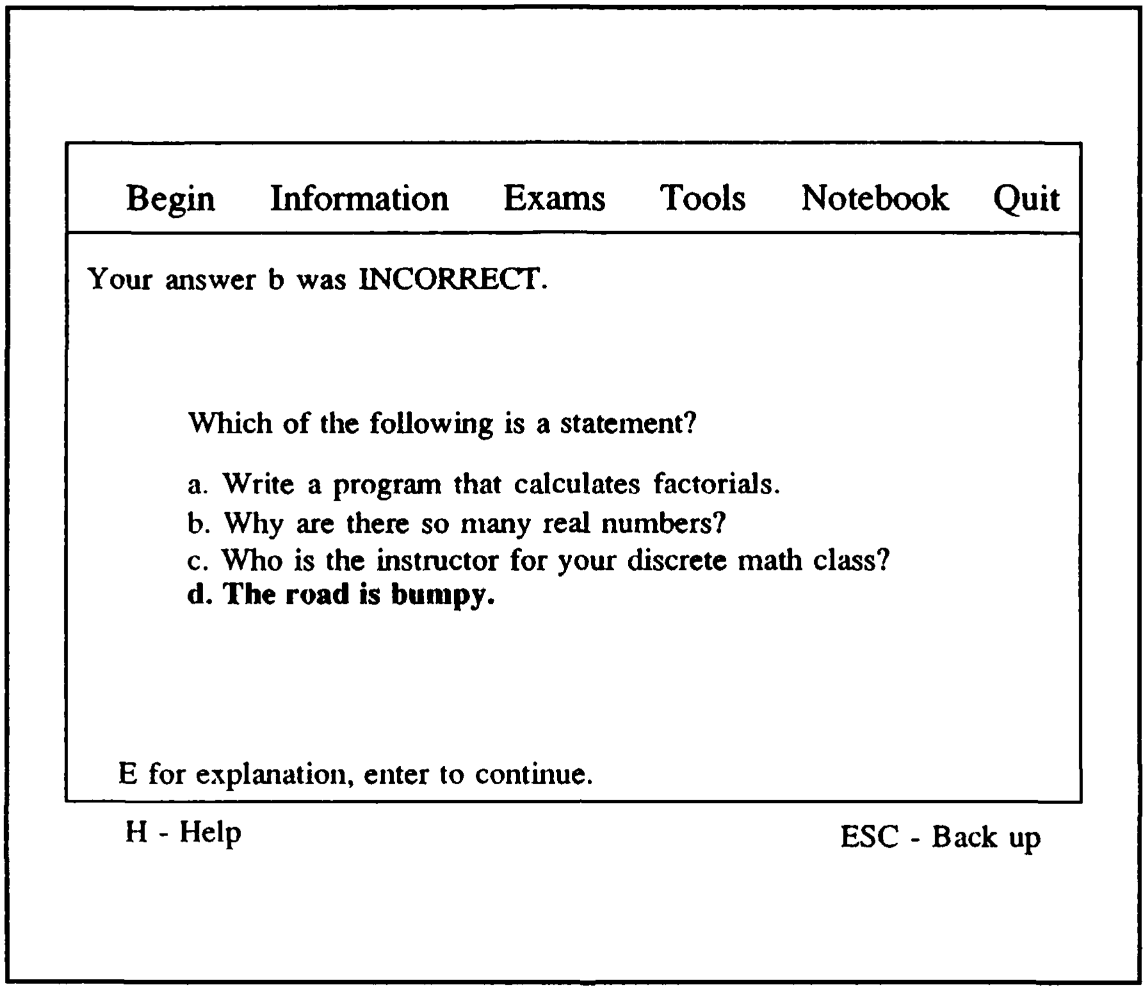
[Enter the Number of Exam Questions Desired]

How many exam questions do you wish? 05

Figure 14 -Selection of Logic Exam

When a particular exam is chosen, the user is asked to enter the number of questions that are desired. The user must respond with a two digit number from 01 to 99. If the user asks for more questions than are available, a message is displayed that shows the total number of questions which are available for that exam. The user is then returned to the lesson.

Once the exam is selected and the number of questions are entered, a random selection process is used to present exam questions to the screen. Questions are answered by typing the letter of the corresponding selection for multiple choice questions and by typing either t or f for true/false questions. After the selection is entered, a message is displayed indicating whether the selection was correct or incorrect. Also, the correct answer is highlighted. If explanations have been provided, typing the *hot key*, *E*, will display an explanation for the corresponding question. Figure 15 shows a question with the solution highlighted. New questions are introduced until the desired number of questions have been presented. After the last question, a *results* screen is displayed.



Your answer b was INCORRECT.

Which of the following is a statement?

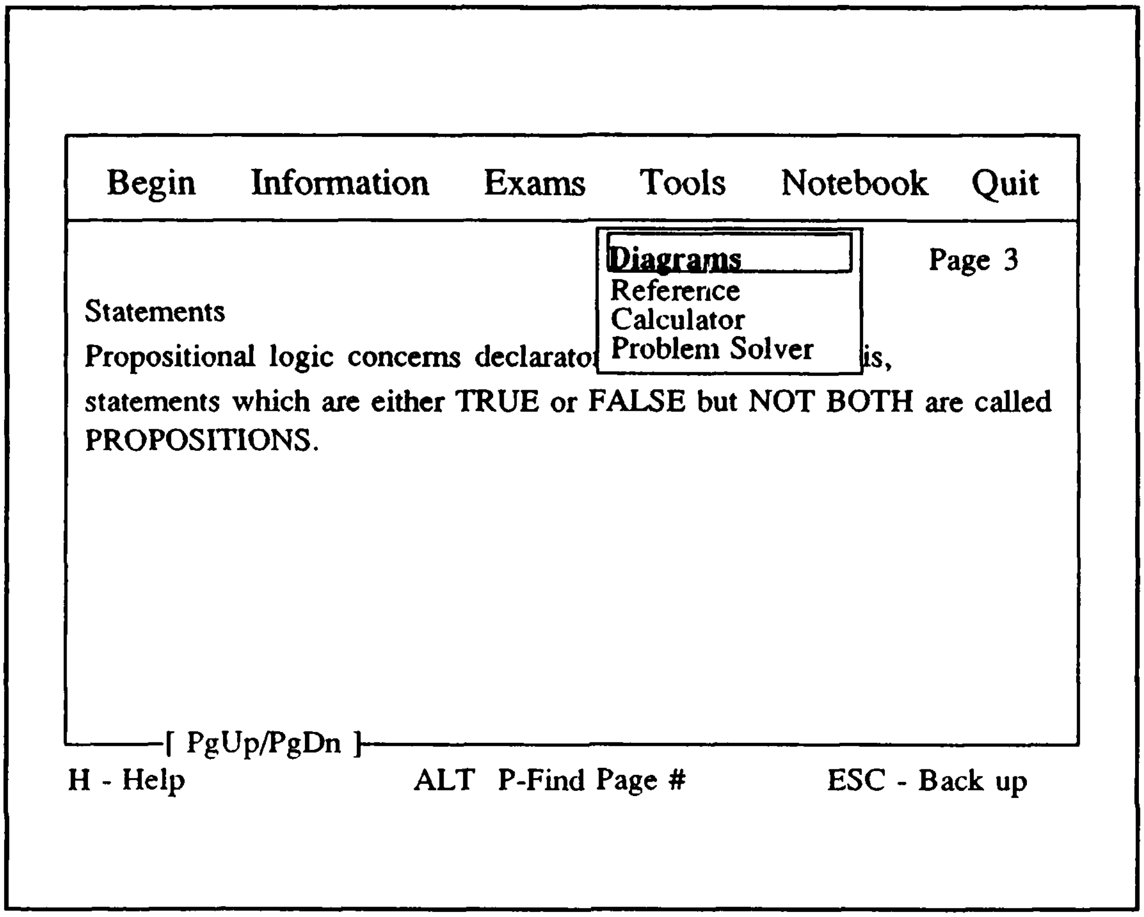
a. Write a program that calculates factorials.
b. Why are there so many real numbers?
c. Who is the instructor for your discrete math class?
d. The road is bumpy.

E for explanation, enter to continue.

Figure 15 -Exam Question with Answer Highlighted

F. TOOLS

Tools, selected from the *main menu* by typing the hot key T, provides the user with instruments that augment the lessons and which aid in the student's understanding of key concepts. Figure 16 shows the interface with a lesson in the background and the *tools* menu selected. From the *tools* menu, *diagrams*, *reference*, *calculator*, or *problem solver* may be selected.

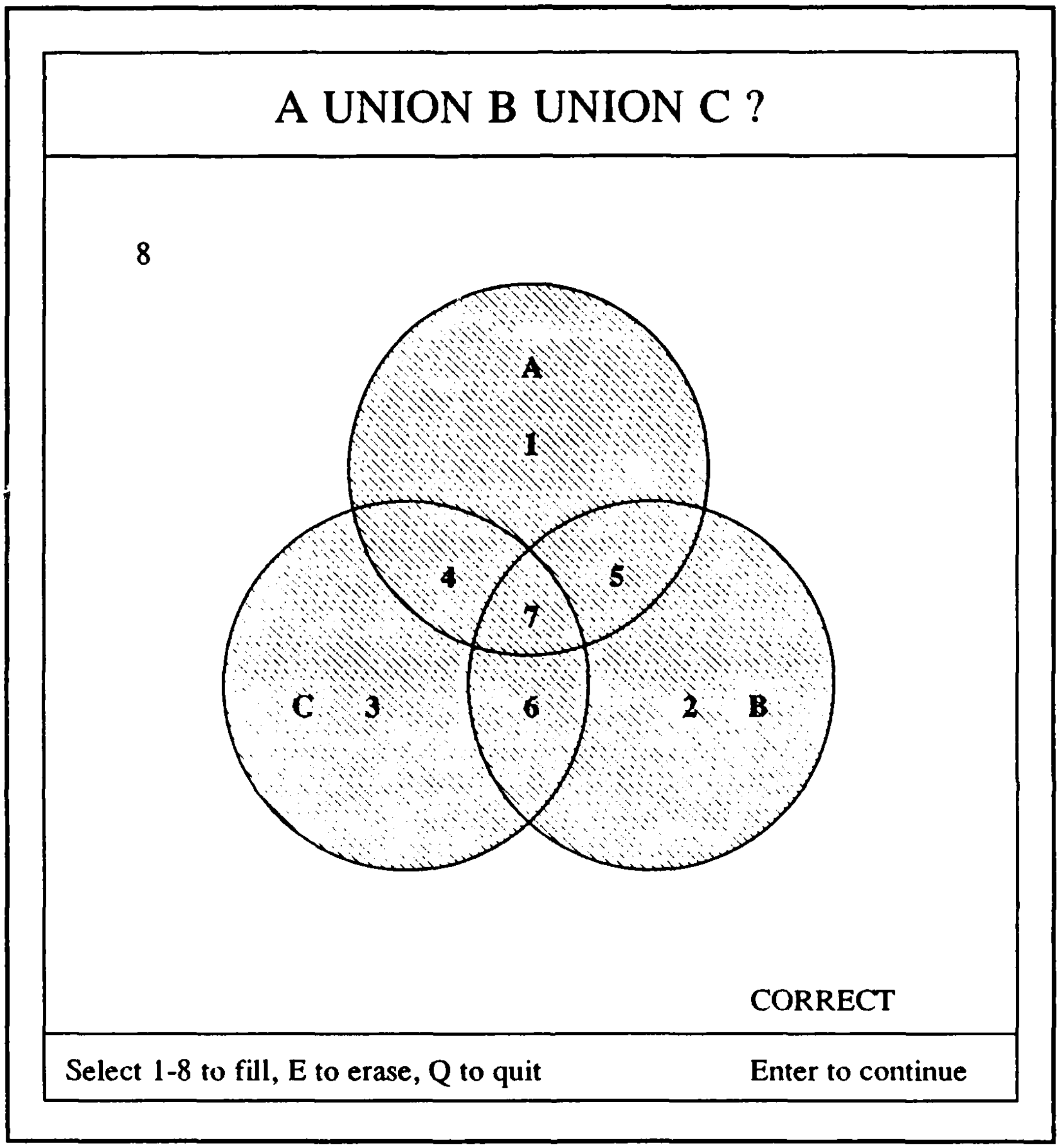


Statements
Propositional logic concerns declarato is,
statements which are either TRUE or FALSE but NOT BOTH are called PROPOSITIONS.

Figure 16 -Selection of the Tools Menu

1. Diagrams

A demanding concept or idea may be presented or practiced pictorially. *Diagrams*, selected from the *tools* menu, are used to graphically rehearse the user. At this time, the *Venn diagram drill* is available through this selection. Figure 17 shows an example of a Venn diagram drill session.



A UNION B UNION C ?

CORRECT

Select 1-8 to fill, E to erase, Q to quit

Enter to continue

Figure 17 -Venn Diagram Drill Session

The *drill session* consists of a randomly generated Venn diagram problem. The problem is displayed at the top of the screen. Three circles are drawn in the center of the screen and represent three sets A, B, and C. The regions of intersection are numbered from one to eight. The bottom of the screen contains instructions and results.

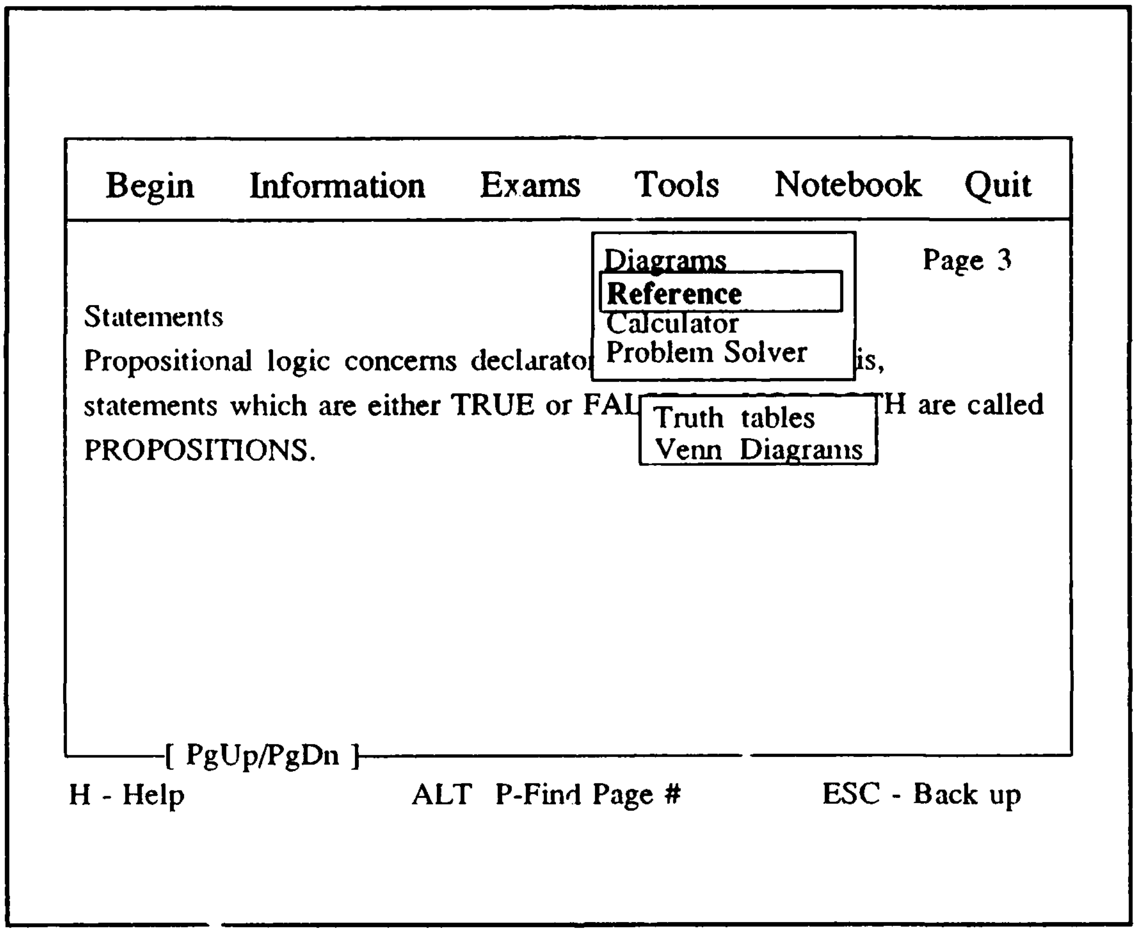
The user is asked to select the number or numbers that correspond to the regions which would be contained in the set of the posed question. As the numbers are selected, the corresponding region is shaded. The user may erase his or her choices and be presented with the original problem by typing the hot key E. Once the region or regions are selected, the user presses the enter key. Then, the user's answer is processed and either CORRECT or INCORRECT is displayed in the bottom right hand corner of the screen. If the answer is correct, the next question is presented after the enter key is pressed. If incorrect, the correct solution is shown before the next question is presented. The user may quit the *Venn diagram drill* and return to the same point in the lesson from which he came by typing the hot key Q.

2. Reference

A quick reference to review key concepts or ideas is made available in this tool. *Reference*, selected from the *tools* menu, is used to rapidly refresh the user's memory in the chosen area. Figure 18 shows that quick reference is available for *Venn diagrams* and *truth tables* and is selected from the *tools* menu.

a. Venn Diagrams

The *Venn diagram quick reference* begins with a menu of available Venn diagram drawings. The user selects the letter corresponding to the desired picture and types enter. Figure 19 displays an example of the resulting drawing. The desired relationship is displayed at the top of the screen. Three circles are drawn in the center of the screen that represent three sets A, B, and C. The area that corresponds to the chosen relationship is shaded. The user may quit the *Venn diagram reference* and return to the lesson by typing the hot key Q.



Statements

Propositional logic concerns declarato is, statements which are either TRUE or FAL H are called PROPOSITIONS.

Figure 18 -Reference Available for Truth Tables and Venn Diagrams

b. Tr:th Tables

There are two choices for *truth table quick reference* and they are *drill* or *rules*. The *rules* section contains a selection of four basic truth tables. The truth tables are chosen by typing one of four function keys; F1, F2, F3, or F4. As shown in Figure 20, the basic truth tables are displayed in a window located at the top right hand corner of the screen. Typing any key other than the four function keys will return the user to the lesson.

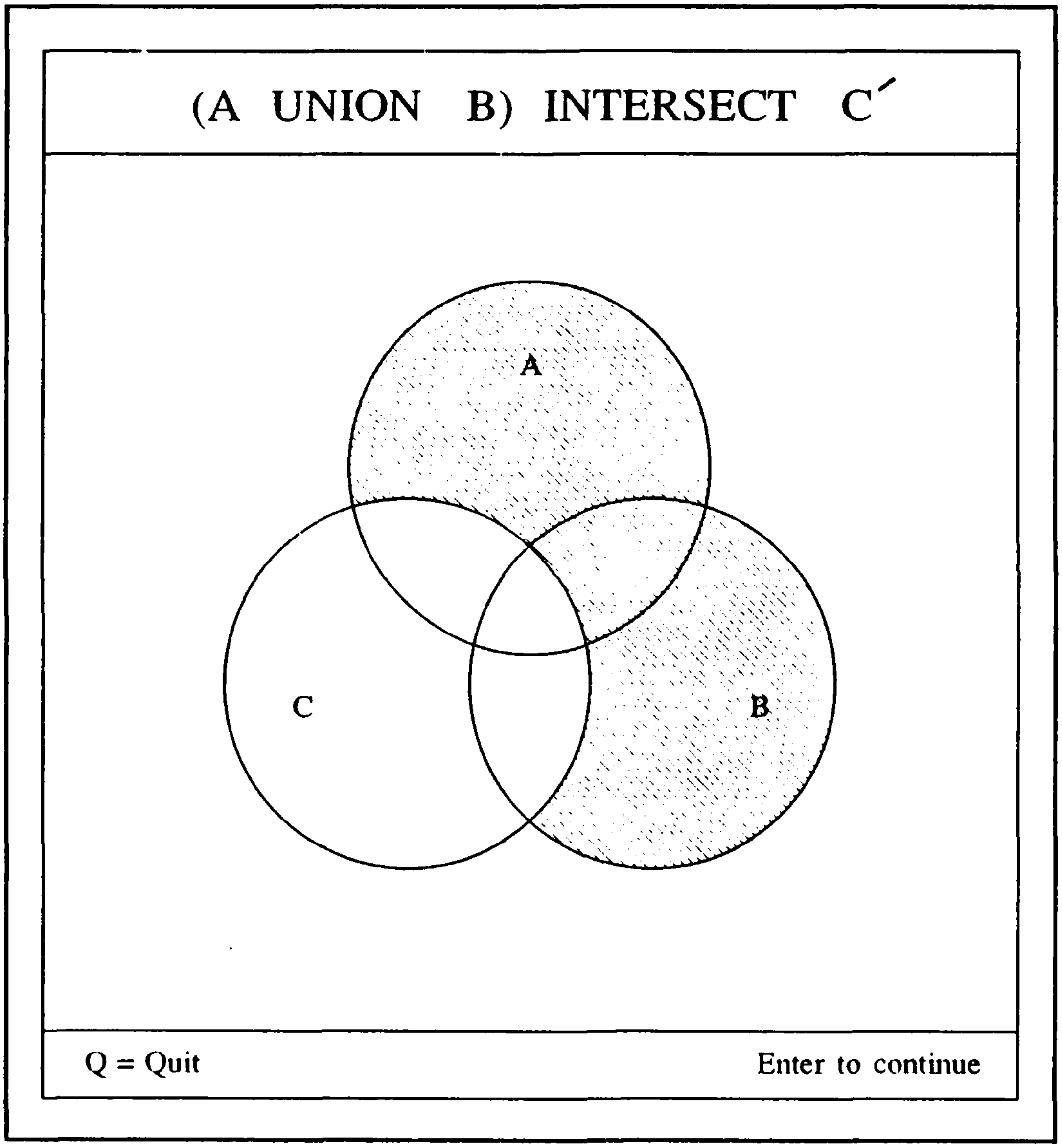
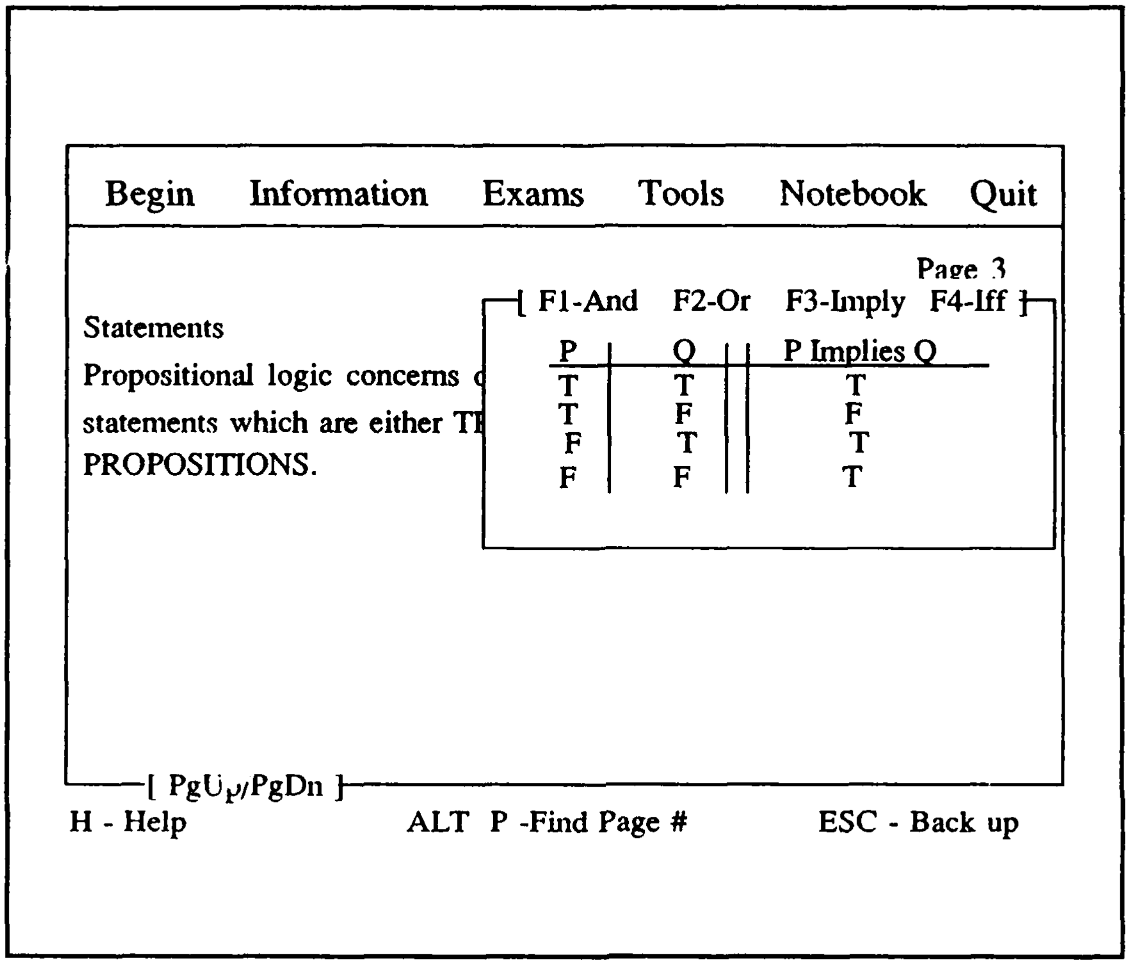


Figure 19 -Sample of a Venn Diagram Quick Reference Drawing

The *drill division* of the *truth table quick reference* includes flash card like practice for the student. An example of the truth table flash cards is illustrated in Figure 21. The user is presented with a randomly selected basic relationship and asked to determine its truth value. Once the user decides on the truth value, he or she types "T" for true or "F" for false. Then, a comparison is made between the given answer and the computed answer. If the user's answer is correct, then "correct" is displayed below the flash card. If incorrect, then "wrong" is displayed. After a short delay, a new flash card is presented and the process is repeated. The student may quit the flash cards by typing Q. After Q is typed, the results of the flash card session are displayed as shown in Figure 22. From the *results* screen, typing any key will return the user to the lesson.



Page 3

Statements

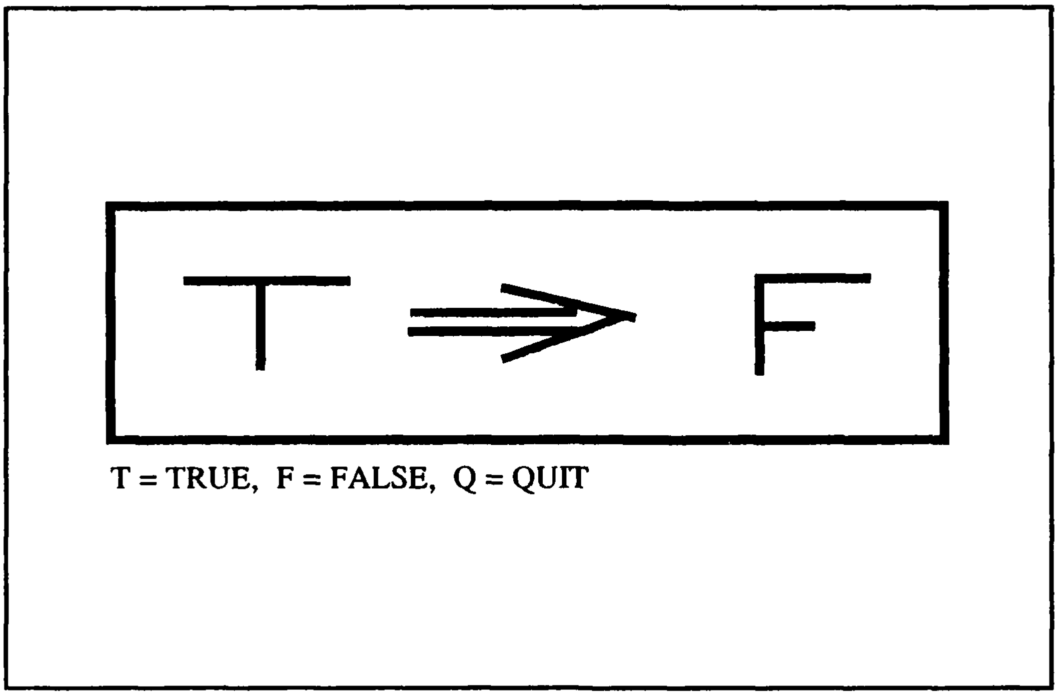
Propositional logic concerns c
statements which are either TI
PROPOSITIONS.

P	Q	P Implies Q
T	T	T
T	F	F
F	T	T
F	F	T

Figure 20 -Display of the Quick Reference Basic Truth Table

3. Calculator

This instrument is available in the *tools* menu (see Figure 16) and is provided so that the user may perform simple mathematical operations without the need of an external calculator. *Calculator* will perform basic addition, subtraction, multiplication, and division. The user may quit the *calculator* by typing the *escape* key.



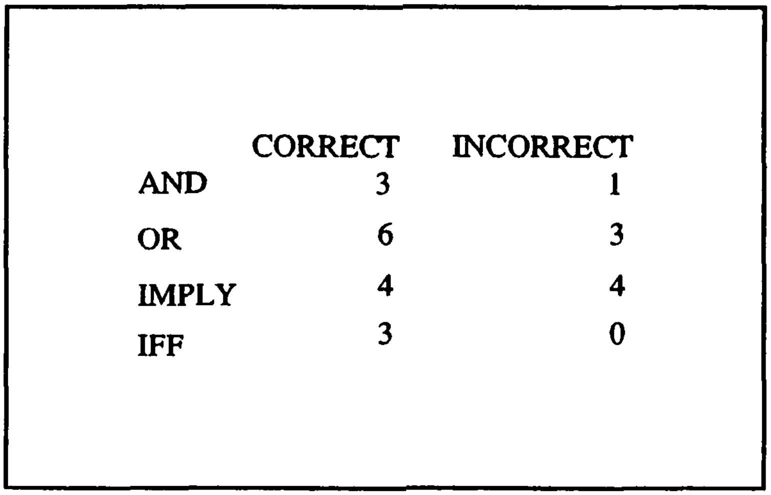
$T \Rightarrow F$

T = TRUE, F = FALSE, Q = QUIT

Figure 21 -Display of Truth Table Flash Card

4. Problem Solver

The *problem solver* is designed to build arbitrary truth tables of moderate size. This tool provides the student with the means to check truth table problems and provides the ability to explore truth tables of his or her own design. The truth table *problem solver* is available in the *tools* menu (see Figure 16).



	CORRECT	INCORRECT
AND	3	1
OR	6	3
IMPLY	4	4
IFF	3	0

Figure 22 -Results Screen for the Flash Card Session

Problems may be explored using four variables (p,q,r,s). The variables may be entered as either upper or lower case, but they are all converted to upper case by the *solver*. That is, Q and q are treated as the same variable, Q, by the solver. The *solver* allows use of five operators which are listed below:

1. ~ (negation),
2. & (and),
3. | (or),
4. > (implication),
5. = (equivalence).

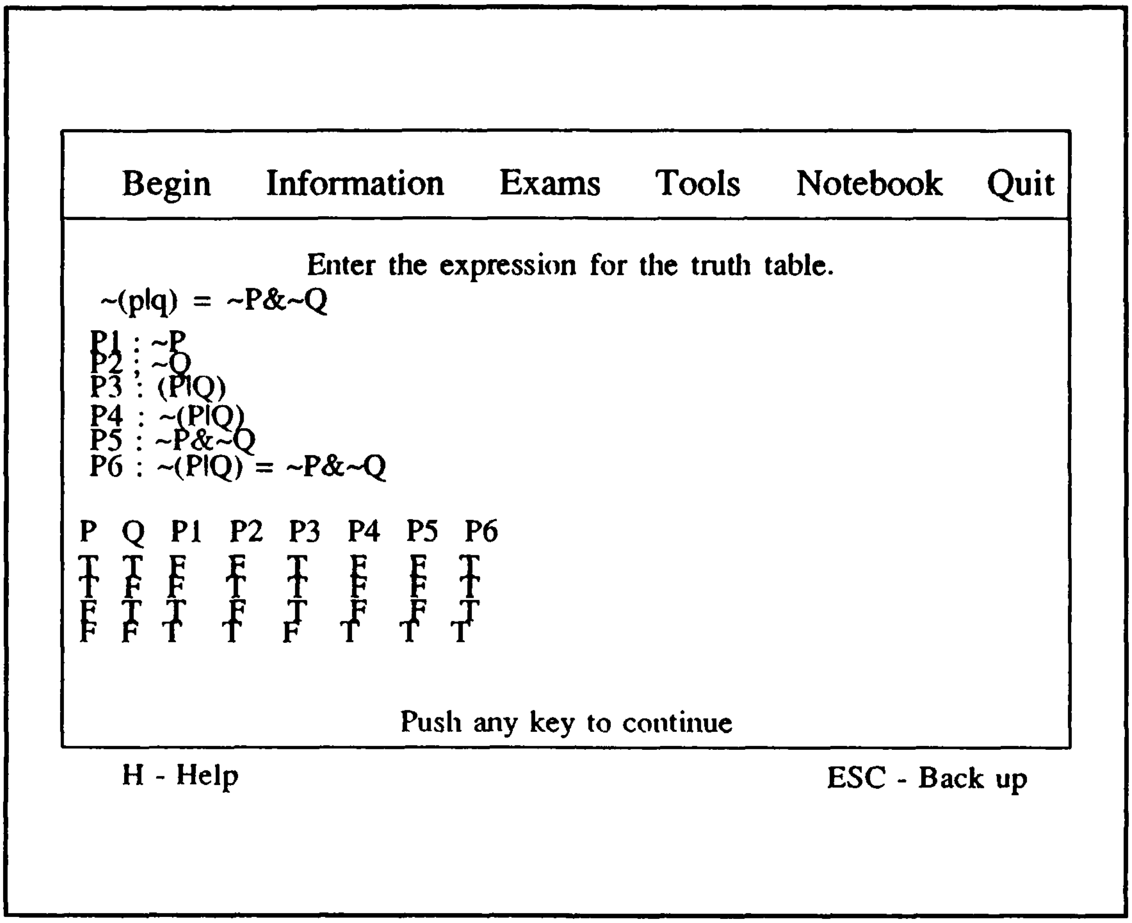
The hierarchy that the *problem solver* obeys is given as follows:

1. negation of variables,
2. operations inside parentheses,
3. negation of operations inside parentheses,
4. and's,
5. or's,
6. implications,
7. equivalences.

Also, operations are executed from left to right.

After the expression is typed onto the screen and the user types *enter (<CR>)*, the *solver* calculates and displays the appropriate truth table. A complete breakdown of the truth table is displayed so that the user may follow the solution step-by-step. The breakdown of propositions is listed above the display of the truth table.

For example, in Figure 23 the expression ~(p|q) = ~P&~Q is investigated. The fourth term evaluated by the solver is P4 and is given as ~(P|Q). The last term, P6, is the originally posed relation. Below the propositions is the truth table. To quit *problem solver* and return to the lesson, the user types the *escape* key.



Enter the expression for the truth table.

~(p|q) = ~P&~Q

P1 : ~P
P2 : ~Q
P3 : (P|Q)
P4 : ~(P|Q)
P5 : ~P&~Q
P6 : ~(P|Q) = ~P&~Q

P	Q	P1	P2	P3	P4	P5	P6
T	T	F	F	T	F	F	T
T	F	F	T	T	F	F	T
F	T	T	F	T	F	F	T
F	F	T	T	F	T	T	T

Push any key to continue

Figure 23 -Example of the Truth Table Problem Solver

G. NOTEBOOK

The *notebook* is a file that contains information that the user deems necessary to isolate for further study. Items listed in the information section may be entered into the notebook. The notebook may be displayed to the screen or sent to the printer for a hard copy. If the notebook is displayed to the screen, the information in the notebook is treated as though it were lesson text. This means that maneuvering inside the notebook is the same as maneuvering inside a lesson (described previously). To quit viewing the notebook the student must select the quit menu with the hot key Q.

Then, from the quit menu, exit is selected. The user is then returned to the same point in the lesson from which he came.

H. QUIT

Prior to quitting the lesson, the user has the option of saving his or her position in the lesson. This option, available in the *quit* menu, is provided so that the user may start the next session on the current page of the present lesson. The *quit* menu, displayed in Figure 24, contains two options. The two options are: *save the current position* and *exit*.

If *save the current position* is selected, the user is prompted to enter his or her social security number. After the ssn is entered, the user is returned to the operating system. For those who do not care to save their last position, the *exit* option may be selected by typing the hot key E. By choosing *exit* from the *quit* menu, the user is immediately returned to the operating system without regard to the present position.

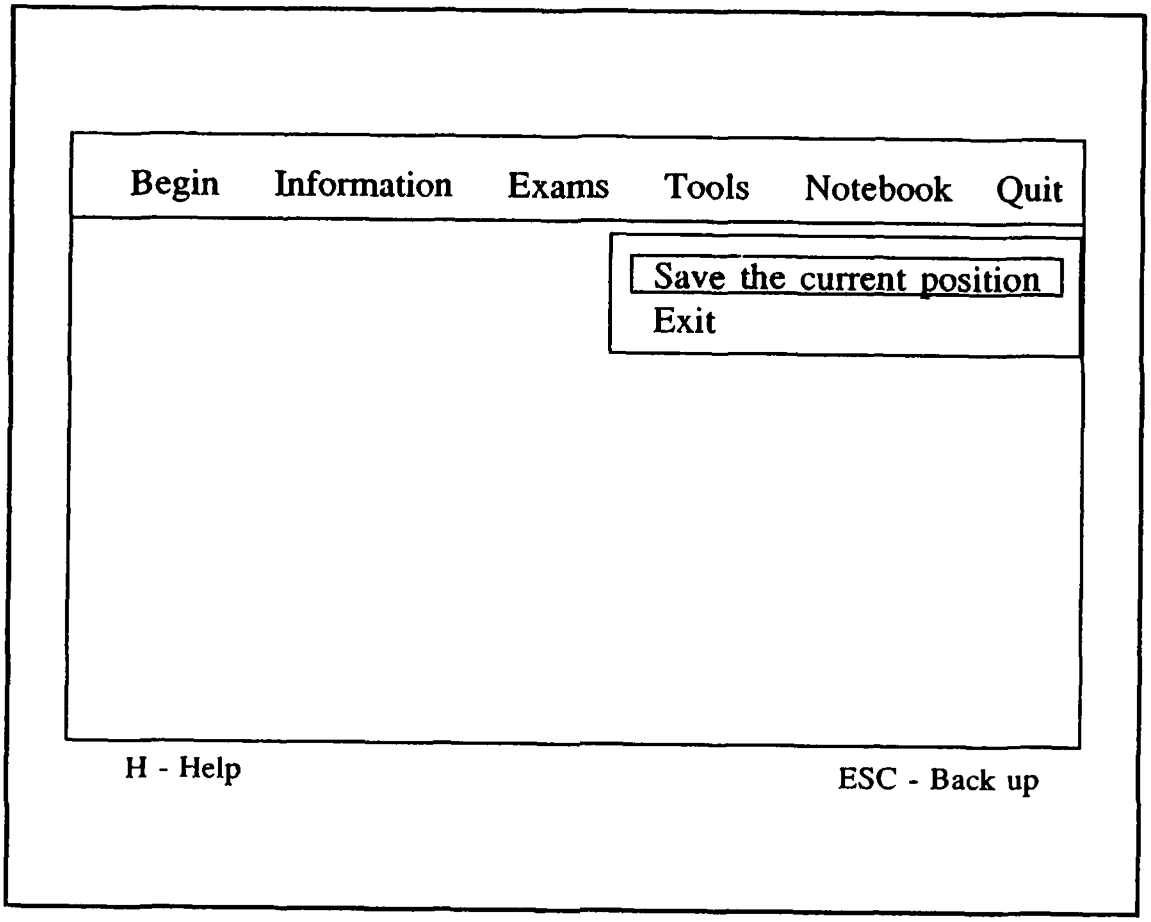


Figure 24 -Quit Menu is Selected

IV. FURTHER WORK

A. INTRODUCTION

The DMT *User Interface* Design Document (See Appendix B) describes the relationships between the four main modules to the system: the *Expert System*, the *Tutor Model*, the *Student Model* and the *User Interface*. Figure 46 of Appendix B shows that the *User Interface* is the main hub for data communications between the four modules. Therefore, implementation of the DMT *User Interface* module before the other three modules is required. This thesis accomplishes that task.

Although the *User Interface* is operational, more work is needed to make the DMT a complete ICAI system. First, testing the *User Interface* on real users will discover the strengths and weaknesses of the design. Second, based on the test results and based on already known extensions, the modification of the *User Interface* will make it more user friendly and more effective. Finally, implementing the other three modules will make the DMT a complete ICAI system.

B. TESTING

The DMT, as it now exists, is simply a prototype. Everything the program does merely shows what is possible. No real lesson in discrete math exists. Therefore, development of complete discrete math lessons is the next important task. Once the lessons are complete, testing can begin on real students.

The testing procedure should answer two distinct questions: (1) how effective is the DMT at teaching the subject domain, and (2) what are the unknown bugs in the program.

Testing the effectiveness of the DMT is not a trivial matter. Testing must evaluate the effectiveness of the interface and the effectiveness of the instruction separately. For instance, it is possible to combine an ineffectual lesson with an effective interface and vice versa. The other possibilities are that the lesson and the interface are both effective or that they are both ineffective. The tester must distinguish between these possibilities and provide ideas on how to improve the interface design or the development of each individual lesson.

Testing for unknown bugs is not an easy matter either. A logical, systematic approach is required to ensure that most of the major program deficiencies are found. When a bug is identified, correcting the bug becomes a priority. If a bug is not correctable, then that bug impacts upon the effectiveness of the interface.

After completion of all tests, the tester must conclude one of two possibilities: the DMT is an effective user interface or it is not.

C. INTERFACE EXTENSIONS

Although the DMT is a working prototype, extensions to the existing software will make the system more user friendly and more effective.

Presently, it is not possible for a non-programmer to modify the interface. All menus are hard coded using Mike Smedley's C Extended Library (CXL) (Smedley,1989). Thus, to add any new lesson to the DMT, a *C language* programmer must physically change the existing DMT code. The major disadvantage in this situation is the time it takes to become familiar with both the CXL functions and the existing DMT code. It is anticipated that most lesson writers for the DMT will not have a programming background. In order to make it easy to add lessons to the DMT, development of an automated menu generation tool is required.

Currently, the DMT does not allow the user any text editing capabilities. Most input from the user is taken from *hot keys* off the keyboard which allows the user to manipulate the menuing system. This type of system lends itself nicely to using a mouse as an input device. Providing mouse support to the user will allow him to *point & shoot* where he needs to go instead of remembering a plethora of unfamiliar keyboard commands. Smedley's CXL package contains functions that support mouse implementation.

One of the original assumptions of this project is that most students interested in this type of instructional software will have access to an AT class computer with an 80286 CPU. Although this assumption is correct today, in five years it may not be true. In recent PC periodicals like BYTE and Dr. Dobbs, the 80286 CPU machine is rarely mentioned. The next generation CPU's like the 80386 and 80486 are the computers that will be available to students in the next decade (Irresistible VGA, 1990), (MAC IIfx, 1990), (Mainstream Amiga, 1990) & (Memory Management, 1990). Therefore, upgrading the DMT to run on one of these machines to take advantage of their unique abilities may increase the effectiveness of the Tutor.

The user's personalized notebook is a key feature to the DMT. It allows the user to store important information for further study. Extensions to the user's interaction with the notebook would greatly enhance the program. One extension might be to add an index page to the notebook that will list each item included and the page number. Another extension will allow the user to edit his notebook while running the DMT program. Other extensions are also possible; but, these two can directly enhance the usability of the notebook as a learning tool.

D. THE NEXT THREE MODULES

As mentioned in the introduction to this chapter, the completion of the DMT *User Interface* represents only 25% of a complete ICAI system. The remaining 75% of the work resides in the unimplemented modules: the *Expert System*, the *Tutor Model* and the *Student Model.* In this thesis, these three modules are referred to as the *Artificial Intelligence (AI)* modules.

The *User Interface* is developed with the Turbo C programming language. Turbo C has been chosen because it is good at manipulating hardware. This is necessary since the *User Interface* is concerned mainly with input/output from the user. Thus, a language that makes it easy to manipulate the input/output devices of the PC is essential.

The remaining *AI* modules require a different programming environment than the *User Interface*. The *AI* modules do not interact directly with the user. Thus, a good hardware manipulation language like Turbo C is not required. Instead, a programming environment that is suitable for implementing *AI* techniques is needed. The only limitation to this environment is that it must have the capability to link with Turbo C executable programs, i.e., the *User Interface*. One language that fits the requirement is Turbo Prolog.

E. WORK LOAD

The proposed extensions and the remaining *AI* modules fit into two basic categories: thesis work and class projects. Also, completion of these extensions require experts in many different fields including computer science, discrete math, C programming and psychology. Figure 25 categorizes each project into the amount of work involved and who should attempt it. Figure 26 is a digraph that details the order in which each project should be attempted.

Topic	Project Type	Student Type
1 Testing/Modification	Thesis Topic	Computer Science
2 Automatic Menu Generation	Class Project	C Programmer
3 Notebook Editing	Class Project	C Programmer
4 Mouse Support	Class Project	C Programmer
5 AT 80386 Upgrade	Thesis Topic	Computer Science
6 Discrete Math Lessons	Class Project	Discrete Math
7 Expert System	Thesis Topic	Computer Science/Discrete Math
8 Student Model	Thesis Topic	Computer Science/Psychology
9 Teaching Model	Thesis Topic	Computer Science/Psychology

Figure 25 -Further Work Summary

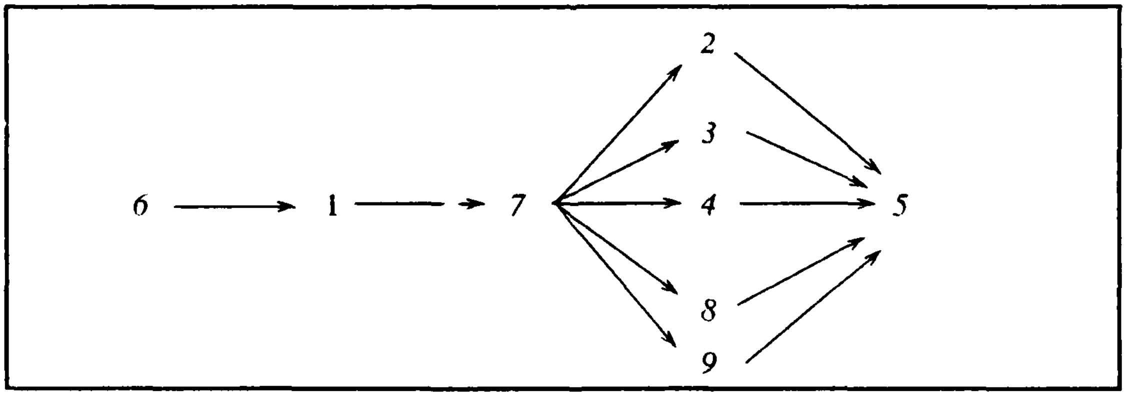


Figure 26 -Order of Future Work

The most difficult problems to solve are the last three listed in Figure 25: the *AI* modules. The *AI* modules are the most difficult because they require experts in two completely different fields to solve each problem. For example, the *Expert System* module requires a computer scientist with a background in artificial intelligence techniques and a mathematician with an emphasis in discrete math. Likewise, the *Student Model* and the Teaching Model both require the same type of computer scientist as the *Expert System* module as well as a psychologist with a background in learning theory.

The combination of computer science and psychology make the *Student Model* and the Teaching Model the most difficult of all. Both topics require the cooperation of two entirely different disciplines. However, both are essential to the successful completion of the DMT. The computer scientist understands how the computer works and the psychologist understands how a student learns.

F. CONCLUSIONS

The Discrete Math Tutor (DMT) is 25% complete. The only module implemented out of the four that make up an Intelligent Computer Aided Instruction (ICAI) system is the User Interface module. However, this phase was not insignificant. The finished User Interface contains 8694 lines of C code and comments (See Appendices E - T).

The work was divided equally between the two authors: Keith Calcote and Rick Howard. Calcote delivered the lesson program and all the tools while Howard designed and implemented the interface. However, both provided insight to each other when problems occurred.

Two key problems were solved in order to make the User Interface work. First, a way to convert ASCII files into a lesson or exam was essential to make the DMT useful. With this functionality, anyone who has access to an ASCII editor may write a lesson or exam that the DMT can easily present. Second, solving the layered memory problem was critical to the future success of the DMT operating as a complete ICAI system. Without the layered memory solution, the User Interface would have exceeded the 640K upper bound of RAM. Consequently, there would not have been any available memory left to add the three unimplimented AI modules.

Notwithstanding, the User Interface is a useful product in its own right. The mixed initiative environment makes the DMT stand out compared to conventional CAI programs found in the public sector. Allowing the student the ability to pursue topics of interest from any point in a lesson provides a necessary measure of student control. The student does not have to sit in front of a computer and read endless screens of text. Instead, he can select what he needs to see and pursue the answers to questions that occur while the lesson is presented.

The DMT is a working prototype. On its own, the DMT demonstrates the potential it has as an effective ICAI system. To make the prototype complete, three tasks must be accomplished:

1. Test the existing DMT for strengths and weaknesses
2. Modify the DMT based on the test results and known extensions
3. Implement the remaining *AI* modules

However, the DMT as it exists now is superior to the standard CAI program and is available right now to teach students certain aspects of discrete math.

APPENDIX A

REQUIREMENTS DOCUMENT

A. INTRODUCTION

The user interface to an intelligent tutor is critical for any ICAI system and is the main focus for this thesis. This appendix describes the mechanics of how the designers envision the user interface for the Discrete Math Tutor (DMT). This document was used as a basis to develop the *Design Document* (See Appendix B) and all code (See Appendices E-T).

B. ENVIRONMENTAL CHARACTERISTICS

1. Minimum Hardware Required

- AT Class Personal Computer with a 8086 CPU or higher.
- EGA graphics card or higher.
- Monochrome monitor or higher.
- A dot matrix printer with draft quality or better.
- 20 Megabyte hard disk or greater.

2. Target Audience

DMT students are assumed to be computer novices with no prior experience in Discrete Math but possessing a strong high school algebra foundation.

C. OVERVIEW

The DMT interface provides the student with the capability to ask any reasonable question that he may have regarding a tutorial lesson. But, the interface is not required to understand the natural human language. The interface also provides a complete learning environment. In other words, when a student sits down to use the DMT, he requires no other materials but the minimum hardware requirements mentioned in Section B1.

The DMT is divided into six sections of functionality:

1. "Begin" and "Quit" are self explanatory
2. "Information" allows the user to review important definitions, theorems and examples pertaining to a lesson of interest.
3. "Pictures" gives the user the capability to construct any type of diagram during any part of the lesson. For example, perhaps a student is trying to solve a Depth First Search problem. The DMT provides the student a means to construct the graph of the problem he wishes to solve.
4. "Algorithms" allows the student to see the results of different algorithms on data that he provides.
5. "Calculator" provides the student a means of determining the answer to quick numerical calculations: addition, subtraction, multiplication and division.
6. "Notebook" allows the student to store key facts such as definitions, theorems, algorithms and examples for future study.

D. STORY-BOARD

1. Opening Screen

Figure 27 is the opening screen to the DMT.

The menu bar at the top represents all the options available for this program.

The DMT highlights the "Begin" portion of the menu bar in a different color initially and moves the highlighted area to any user selected option.

When one of the above menu options is selected, the DMT displays a pop-up menu that lists further choices.

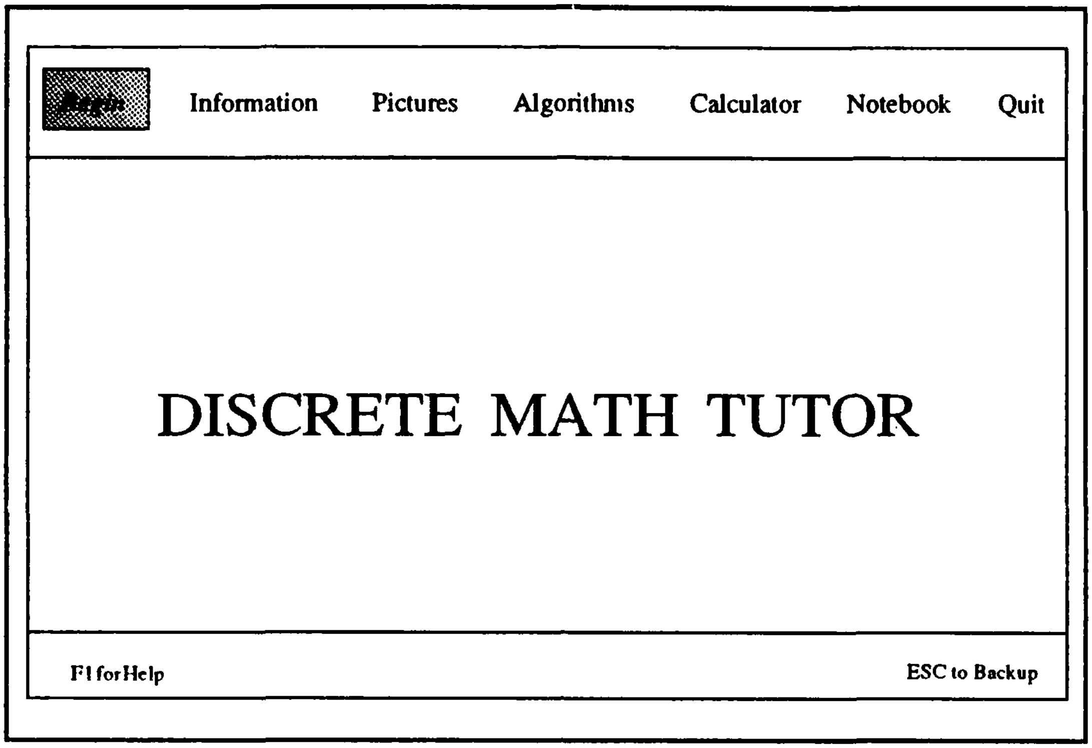


Figure 27 -Opening Screen

2. "Begin" Pop-up Menu

The menu shown in Figure 28 is presented if *Begin* is chosen from the top menu bar. The DMT displays other pop-up menus depending on the choice made here.

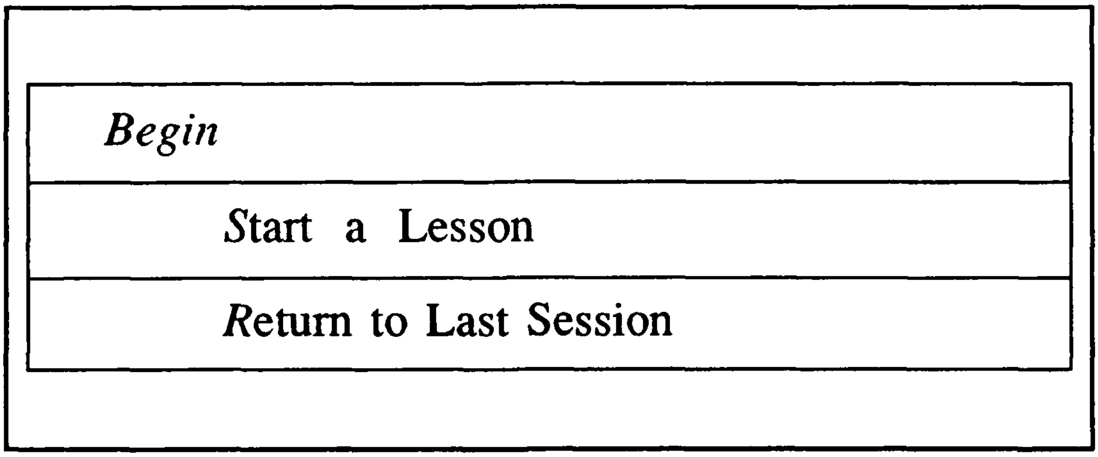


Figure 28 -"Begin" Pop-up Menu

3. "Start a Lesson" Pop-up Menu

The menu shown in Figure 29 is presented if *Start a Lesson* is chosen from the "Begin" Pop-up menu. The user chooses the lesson he wishes to study. The lessons listed here are just examples and do not reflect what will actually be included in the DMT. Each lesson is listed in the recommended sequence of study.

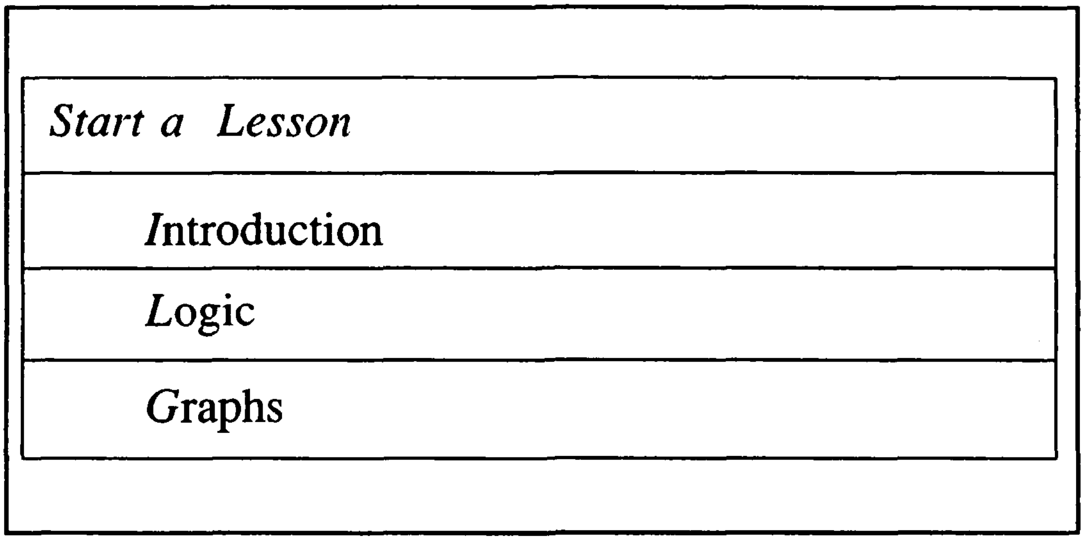


Figure 29 -"Start a Lesson" Pop-up Menu

4. "Return to Last Session" Pop-up Menu

The menu shown in Figure 30 is presented if *Return to Last Session* is chosen from the "Begin" Pop-up menu. The user enters his social security number. The DMT then displays the user's last screen in his previous session.

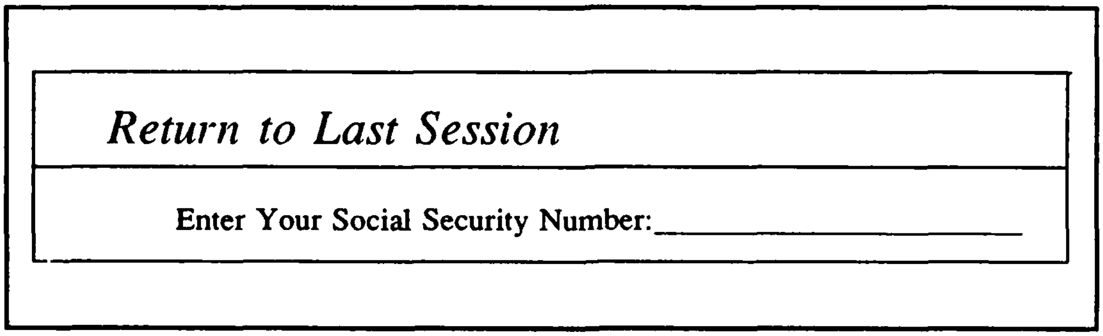


Figure 30 -"Return to Last Session" Pop-up Menu

5. "Information" Pop-up Menu

The menu shown in Figure 31 is presented if *Information* is chosen from the top menu bar.

When one of the options is selected, the DMT displays a pop-up menu that lists further choices.

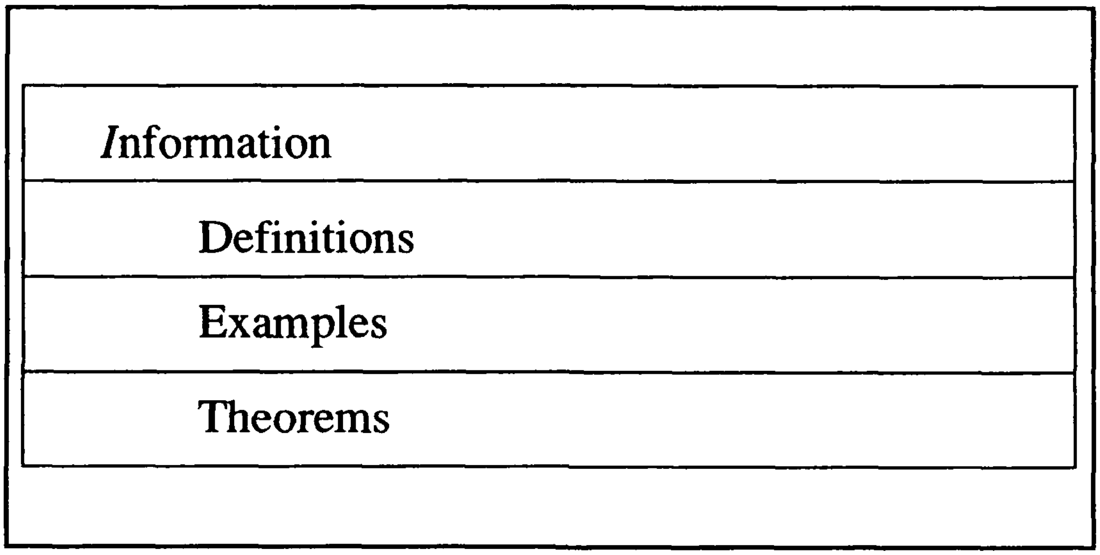


Figure 31 -"Information" Pop-up Menu

6. "Definitions" Pop-up Menu

The menu shown in Figure 32 is presented if *Definitions* is chosen from the "Information" Pop-up menu. It lists the names of all definitions in the DMT by name in alphabetical order. After the user chooses the definition he wishes to review, the DMT displays the entire definition in a pop-up window. The DMT then presents the options shown in Figure 33.

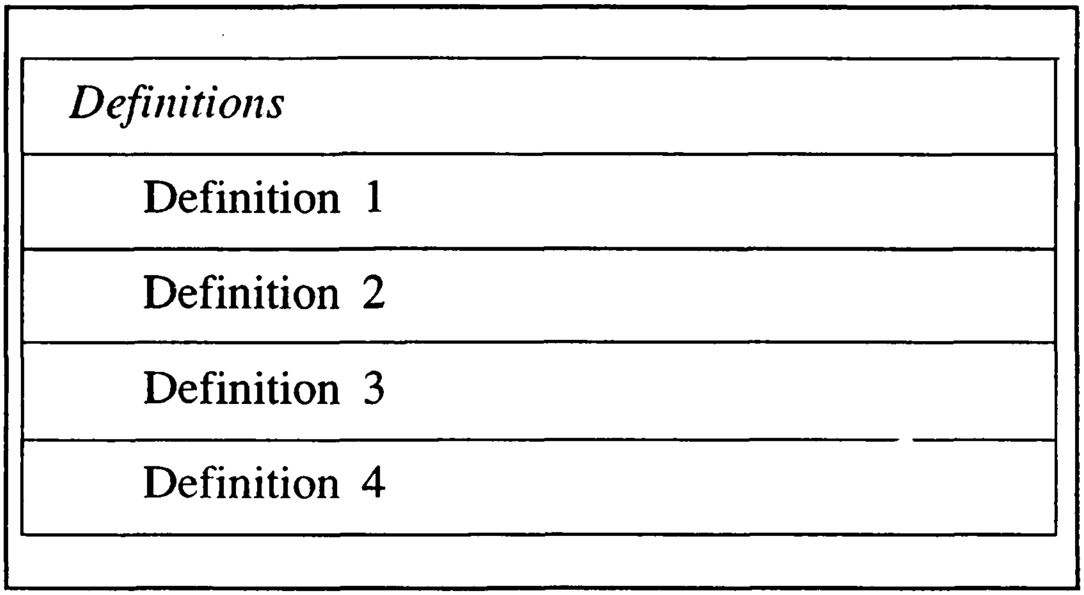


Figure 32 -"Definitions" Pop-up Menu

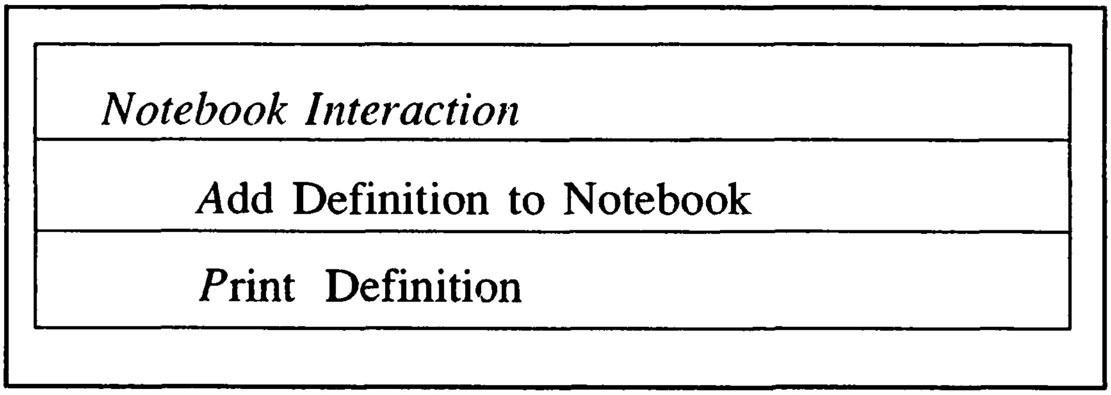


Figure 33 -"Notebook Interaction" Pop-up Menu

If the user chooses "Add Definition to Notebook", the DMT concatenates the entire definition to the end of the student's notebook .

If the user chooses "Print Definition", the DMT outputs the definition to the printer.

7. "Examples" Pop-up Menu

The menu shown in Figure 34 is presented if *Examples* is chosen from the "Information" Pop-up menu. It lists the names of all examples in the DMT by name in alphabetical order. After the user chooses the example he wishes to review, the DMT displays the entire example in a pop-up window.

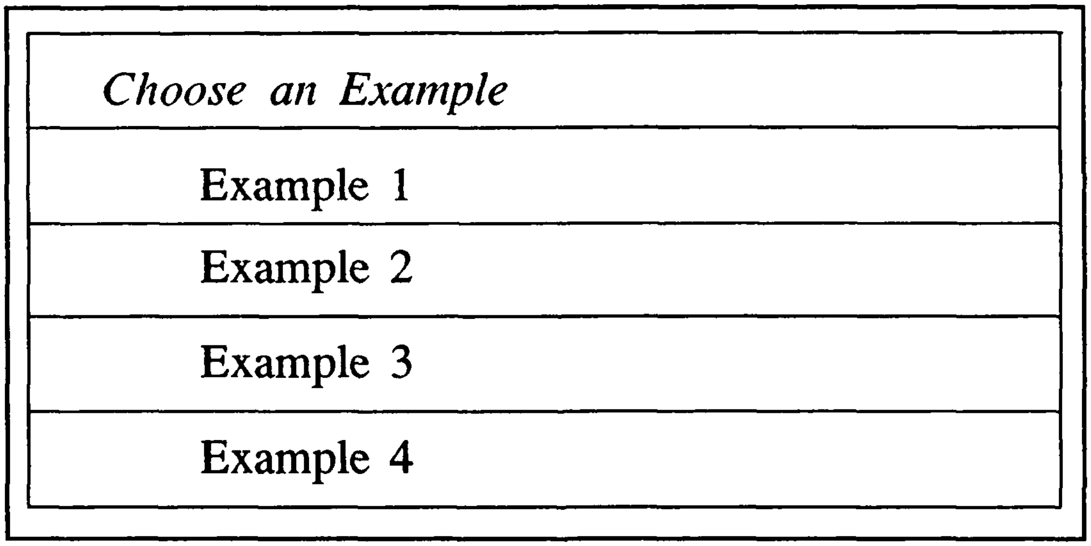


Figure 34 -"Choose an Examples" Pop-up Menu

The DMT then presents the menu shown in Figure 35.

If the user chooses "Add Example to Notebook", the DMT concatenates the entire example to the end of the user's notebook.

If the user chooses "Print the Example", the DMT outputs the example to the printer.

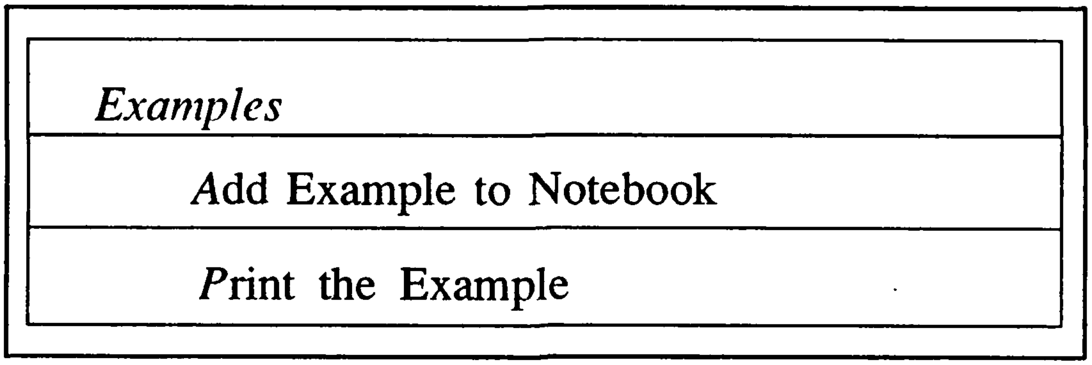


Figure 35 -Examples

8. "Theorems" Pop-up Menu

The menu shown in Figure 36 is presented if *Theorems* is chosen from the "Information" Pop-up menu. It lists the names of all theorems in the DMT by name in alphabetical order. After the user chooses the theorem he wishes to review, the DMT displays the entire theorem in a pop-up window. The DMT then presents options shown in Figure 37.

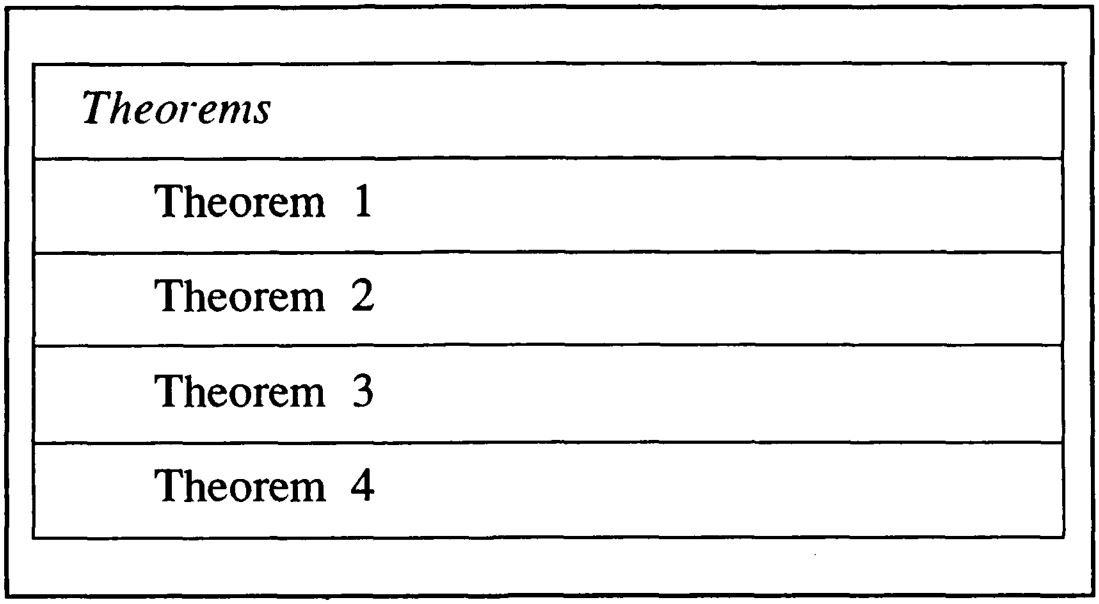


Figure 36 -"Theorems" Pop-up Menu

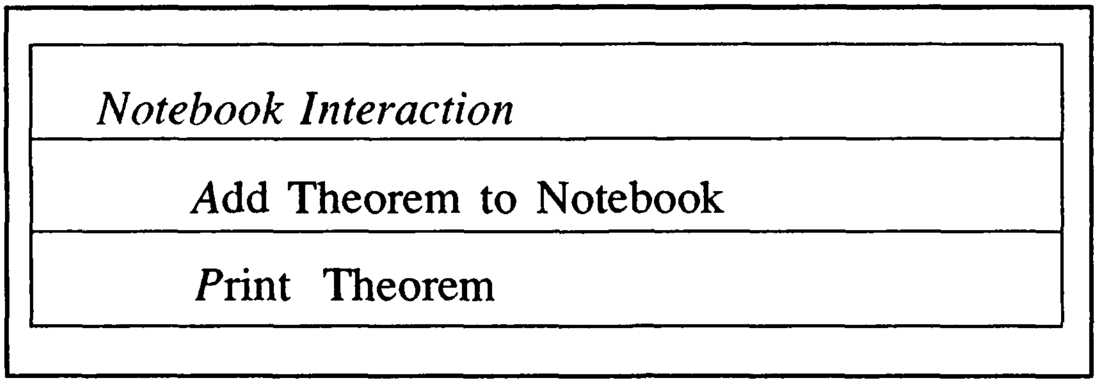


Figure 37 -"Notebook Interaction" Pop-up Menu

If the user chooses "Add Theorem to Notebook", the entire theorem is concatenated to the end of the user's notebook.

If the user chooses "Print Theorem", the DMT outputs the theorem to the printer.

9. "Pictures" Pop-up Menu

The menu shown in Figure 38 is presented if *Pictures* is chosen ırom the top menu bar. Choosing an option from this menu allows the user to draw a picture in a pop-up window. The Truth Table and the Graph are just examples. It is not known what kind of pictures are required. This determination will depend upon how the authors of any particular lesson construct their lesson plan. However, if a picture of some sort is required in any lesson, the DMT will provide the user the drawing capability from this menu selection.

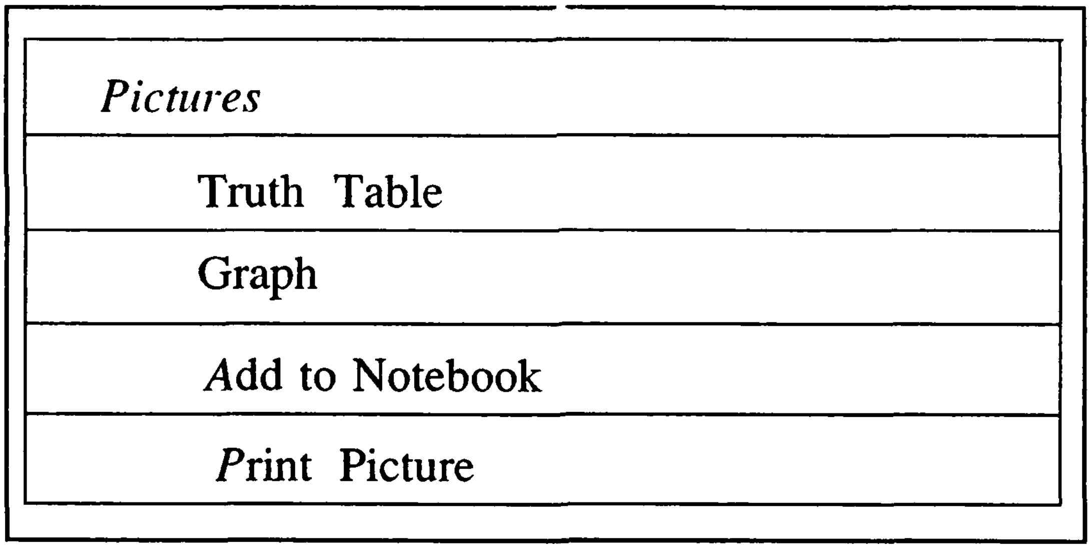


Figure 38 - "Pictures" Pop-up Menu

10. "Algorithm" Pop-up Menu

The menu shown in Figure 39 is presented if *Algorithm* is chosen from the top menu bar. It lists the names of all algorithms in the DMT by name in alphabetical order. After the user chooses the algorithm he wishes to review, the DMT presents the options shown in Figure 40.

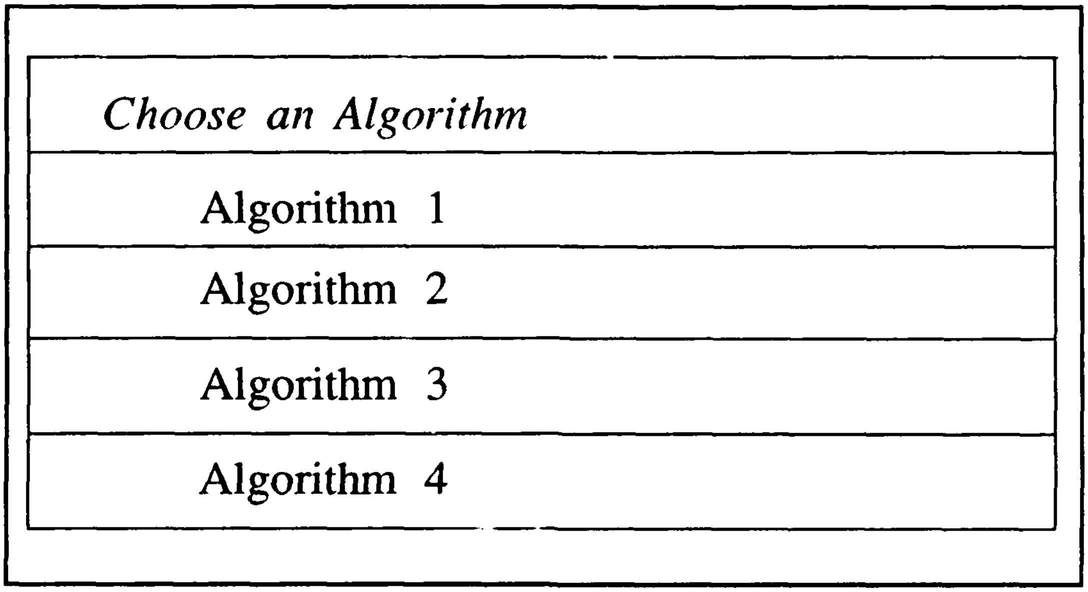


Figure 39 -"Choose an Algorithm" Pop-up Menu

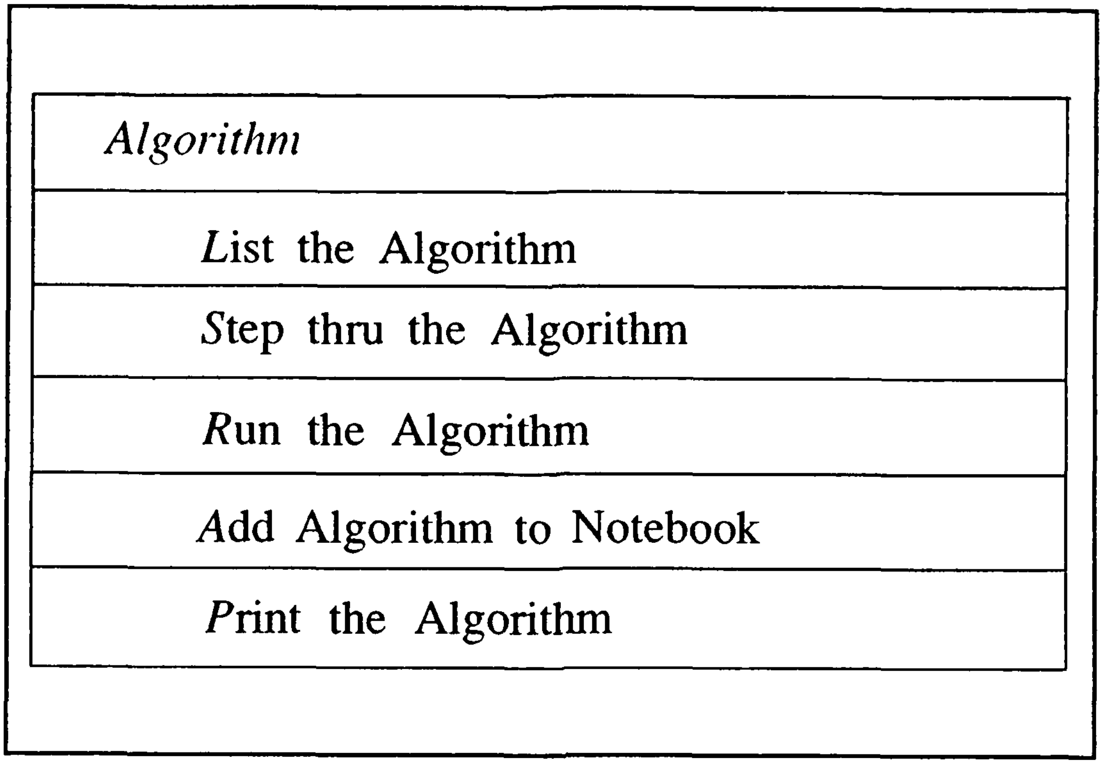


Figure 40 -"Algorithm" Pop-up Menu

If the user chooses "List the Algorithm", the DMT displays a description of the algorithm in a pop-up window.

If the user chooses "Step thru the Algorithm", the DMT walks the user thru the chosen algorithm one step at a time in a pop-up window on data that the user provides.

If the user chooses "Run the Algorithm", the DMT displays the answer to a set of user provided data in a pop-up window using the chosen algorithm.

If the user chooses "Add Algorithm to Notebook", the DMT concatenates the entire algorithm to the end of the user's notebook.

If the user chooses "Print the Algorithm", the DMT outputs the algorithm to the printer.

11. "Calculator" Pop-up Window

The pop-up window shown in Figure 41 is presented if *Calculator* is chosen from the top menu bar. The user enters calculations into the calculator in "INFIX" notation straight from the keyboard. For example, if the user wishes the results of the addition 2+2, he enters the following data:

2+2=

The DMT shows the "2+2" in the display portion of the calculator. When the user pushes "=", the DMT clears the display and presents the answer.

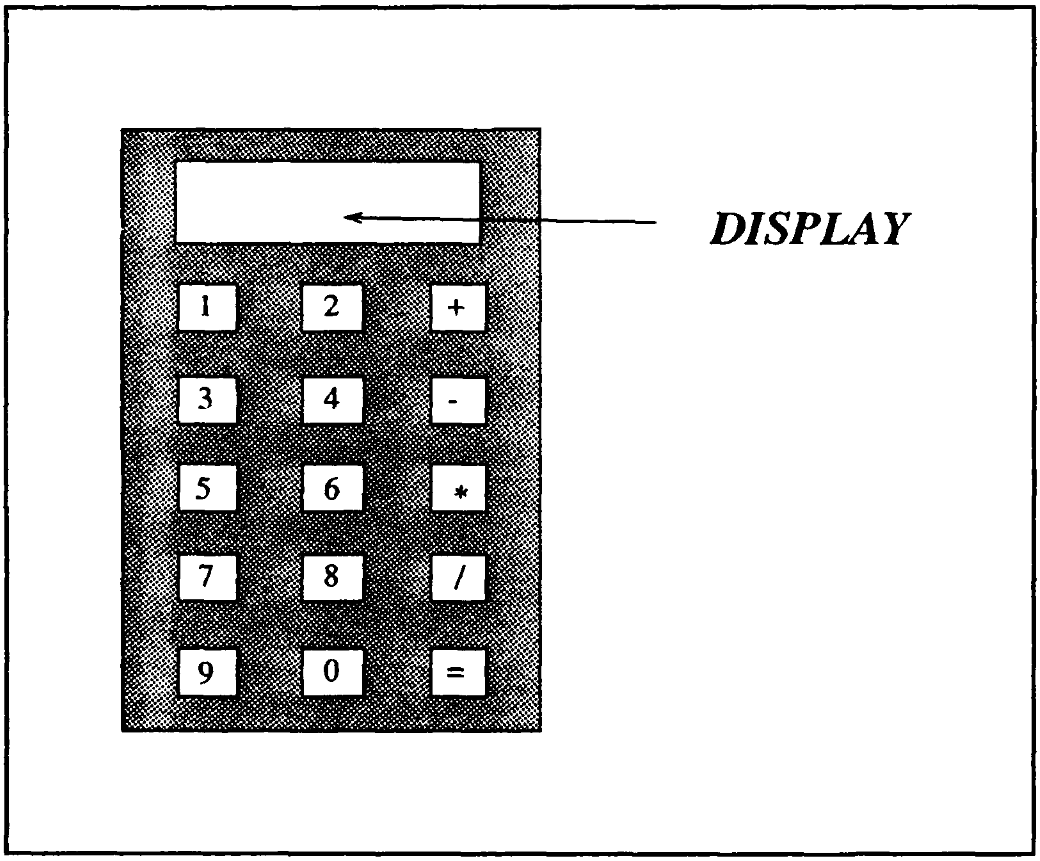


Figure 41 -"Calculator" Pop-up Window

12."Notebook" Pop-up Menu

The menu shown in Figure 42 is presented if *Notebook* is chosen from the top menu bar.

If the user chooses "View Notebook", the DMT displays a pop-up window with the contents of the user's personal notebook. The DMT allows the user to page up and down his notebook. The user presses the escape key to return to his previous window.

If the user chooses "Print Notebook", the DMT sends the contents of the user's personal notebook to the printer.

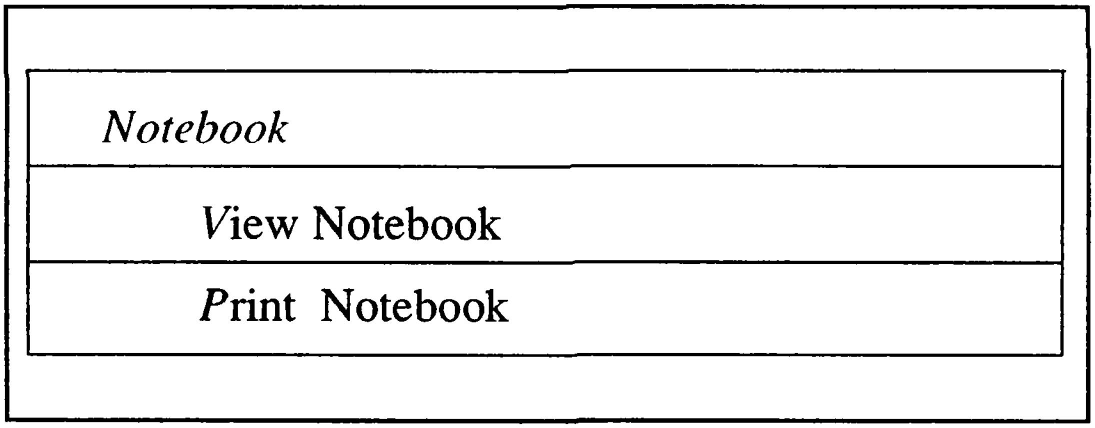


Figure 42 -"Notebook" Pop-up Menu

13. "Quit" Pop-up Menu

The menu shown in Figure 43 is presented if *Quit* is chosen from the top menu bar.

If the user chooses "Save Current Position", the DMT presents the same display as presented in Section 3D: Return to Last Lesson. After the user enters the correct SSN, the DMT returns the user to the "Quit" pop-up menu. If the user used "Return to Last Session" upon the start of this session, the DMT will not make the user enter the SSN again. It will remember the current SSN and store the current lesson session position under that SSN. This functionality allows the user to return to his last location in the DMT when he returns.

If the user chooses "Display Results", the DMT will display the student's performance in a pop-up window. The "Display Result" screen is not yet determined since the student model has not been designed.

If the user chooses "Exit", the program terminates.

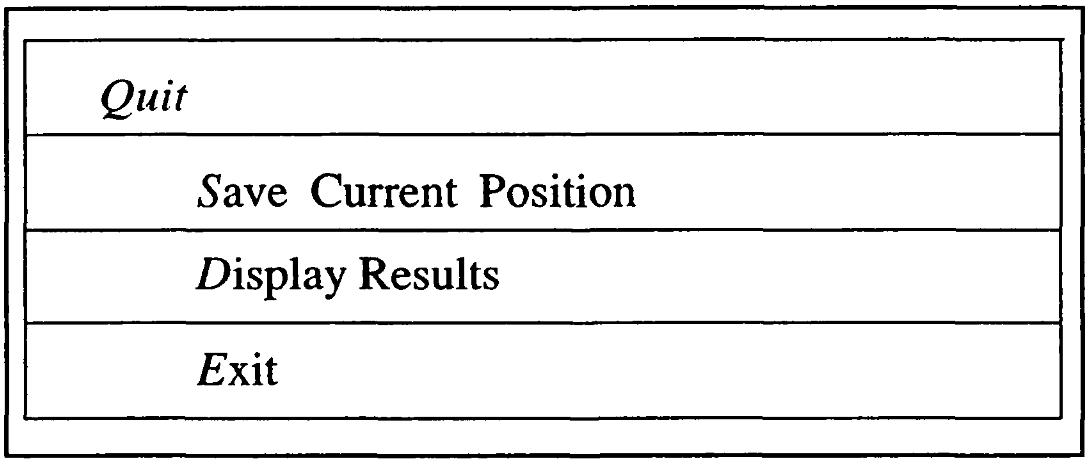


Figure 43 -"Quit" Pop-up Menu

14. "Help" Pop-up Menu

Help is not a selection from the top menu bar. However, it is constantly available to the user if he presses the F1 key. Help presented to the user is context sensitive. This means that the DMT will present an appropriate help screen no matter where the user is in the system.

15. General Notes

On any pop-up menu, the DMT will not accept keystrokes that are not valid choices.

All pop-up menus stay on the screen until a choice is made. An example is shown in Figure 44.

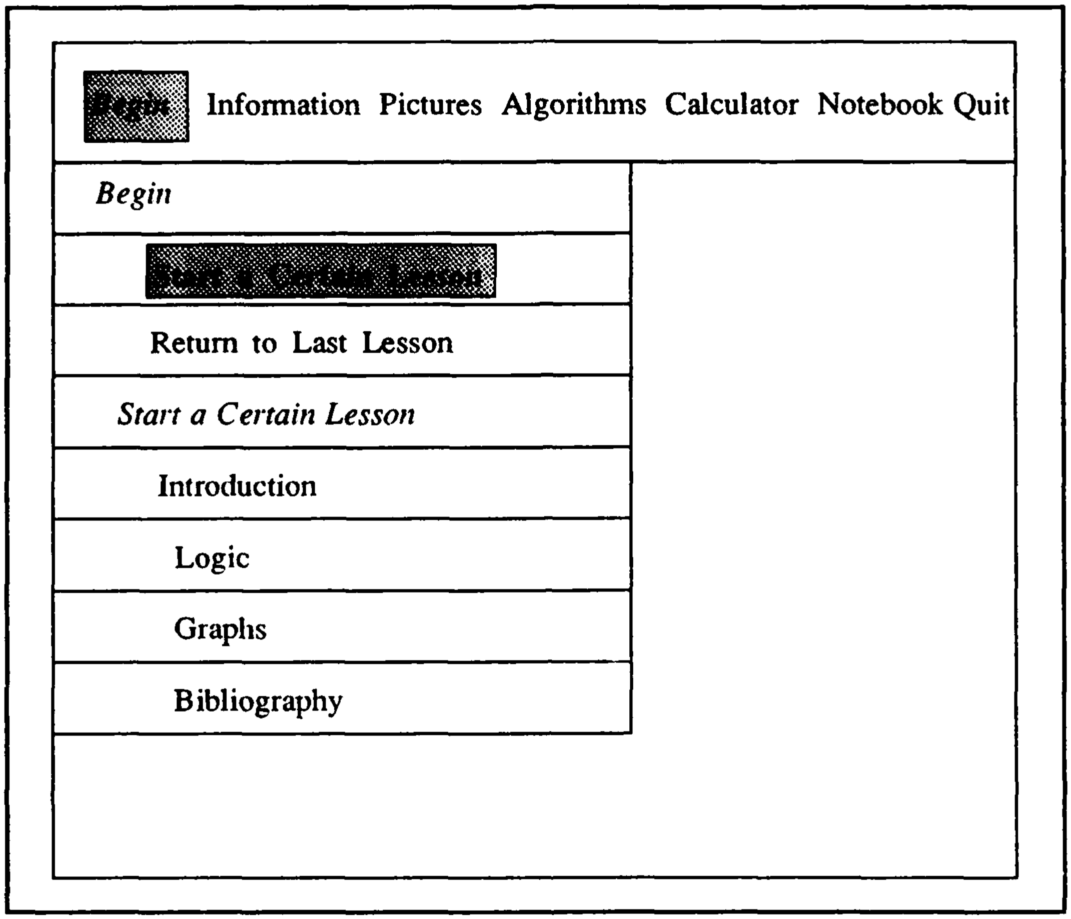


Figure 44 -Cascading Menu Example

The user chose the "Begin" pop-up menu. The DMT offeres two choices: Start a Lesson and Return to Last Session. Suppose the user chose Start a Lesson. The DMT then displays the "Start a Lesson" pop-up menu.

After the user chooses which lesson he wants, the DMT erases the pop-up menus and starts the chosen lesson.

The user chooses menu items in one of three ways: arrow keys and a carriage return, key letters in the menu choice (indicated by italicized first letters in all menus), or the point and shoot mouse.

The escape key allows the user to return to the previous window or pop-up menu at any time.

E. CONSTRAINTS & GOALS

1. Constraints

Lesson authors must write lessons with the DMT interface in mind. After a lesson is developed, the author must prepare the following additional screens: definition, algorithm, theorem, and example.

Any diagrams needed in the lesson must be provided in the "Pictures" portion of the DMT.

Any new calculator functions needed by the students must be added to the "Calculator" portion of the DMT.

2. Goals

The interface allows the user to ask any pertinent question about the lesson without having the computer act as a language interpreter.

The DMT is an evolving system. As new lessons are added, new functionality to the DMT interface must be added.

The DMT Interface is an overlay to any math tutor system. In other words, the DMT interface could be laid on top of any existing math tutor as long as the constraints mentioned above were met. This means that the DMT interface could become a standard for any math tutor.

F. LIFE CYCLE CONSIDERATIONS

The desired DMT should function as described.

If problems occur in the design and implementation phase of this project, the following reduced DMT functionality will be implemented:

- The "Algorithm" portion of the DMT will only run algorithms and not allow the user to step thru an algorithm.
- The "Notebook" portion of the DMT will not allow the user to page up and down the contents of the notebook. Instead it will allow the user to scroll one way through the notebook.

If no problems occur during the design and implementation phase of this project, the following functionality will be added to the DMT:

- Add pages to the student notebook so that all entries can be indexed.
- Allow the user to selectively erase portions of the notebook.

APPENDIX B

DESIGN DOCUMENT

A. INTRODUCTION

The user interface to an intelligent tutor is critical for any ICAI system and is the main focus for this thesis. This appendix describes the design of the user interface for the Discrete Math Tutor (DMT) using the Software Design methodology. This document was completed after the *Requirements Document* (See Appendix A) and prior to any written code (See Appendices E-T).

B. SOFTWARE DESIGN METHODOLOGY (SWD)

The major component to SWD is the Data Flow Diagram or DFD. The DFD is used to model any type of system. SWD uses only four symbols to describe a DFD:

1. External Entity □ -> The source/destination of data outside the system.
2. Data Flow — -> A path that data follows.
3. Process ▭ -> A function that transforms data.
4. Data Store ⊐ -> A place to store data.

From the DFD, the Data Dictionary is derived. The Data Dictionary represents an abstract view of the type of information inside the system. (Gane, 1978)

C. SWD APPLIED TO THE USER INTERFACE

1. The DFD

SWD is a hierarchical procedure. The designer begins with the "Big Picture" abstract view of his application and uses a DFD to describe it.

Figure 45 is a simple, abstract view of the Discrete Math Tutor (DMT). Note that the user is an outside entity to the DMT. He provides "Input" data and "Problems" data to it. He receives "Solutions" data and "Lessons" data from it. These data items become the first entries listed in the DMT Data Dictionary.

More detail to this level of abstraction is added by developing a DFD for each "Process" described. For example, a DFD for the DMT process is shown in Figure 2.

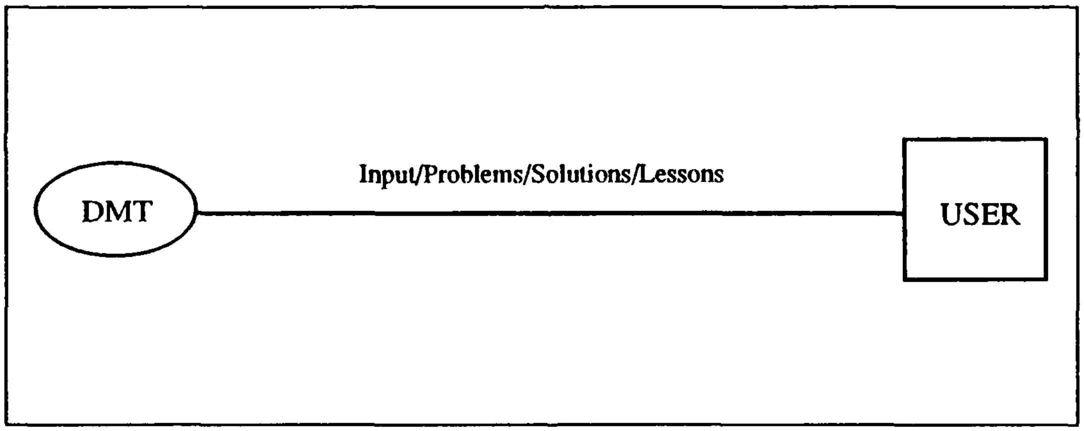


Figure 45 -SWD Top Level abstraction of the Discrete Math Tutor (DMT)

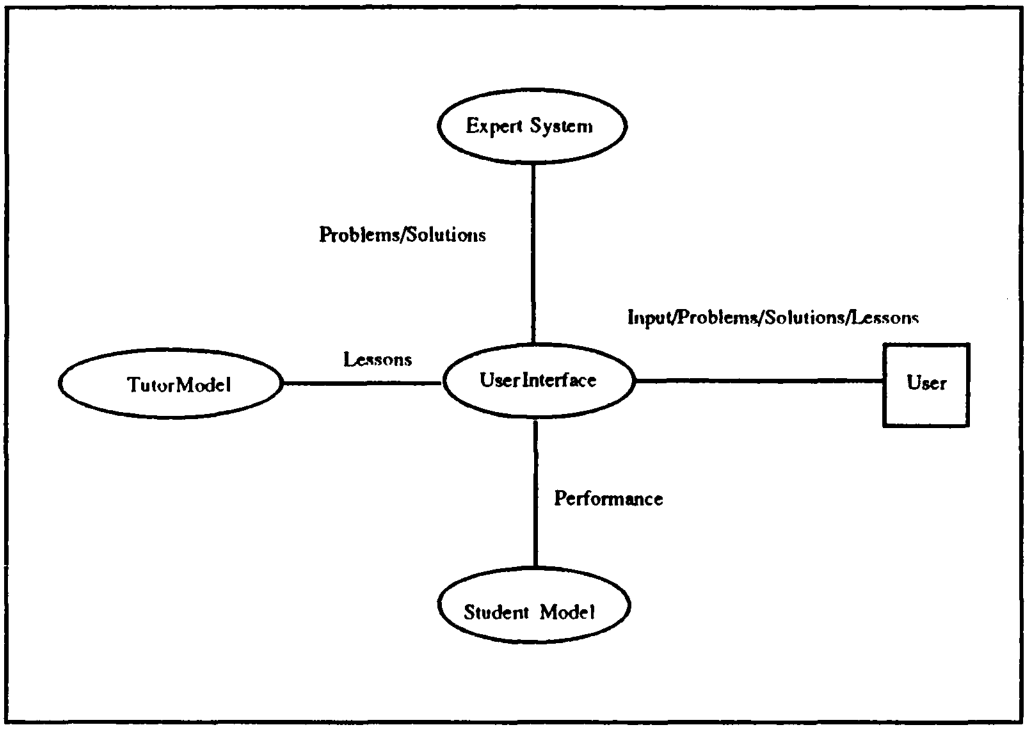


Figure 46 -DFD for the DMT Process

In Figure 46, the four basic modules described in the thesis introduction appear for the first time. A new entry into the Data Dictionary includes the "Performance" data item.

Further detail is now added to this level of abstraction by following the same procedure. Since this thesis deals with the user interface, the next abstraction level concerns the DFD for the User Interface process and is shown in Figure 3.

Notice that in Figure 47, the *Tutor Module, Expert Module, Student Module* and *User* are all external entities to the *Interface Processing System;* however, the Information Processing System receives the same data items described in Figure 46. Further, Figure 47 shows the Data Store symbol for the first time. These Data

Stores hold information in the form of Text for use by the *Interface Processing System*. Thus, the Data Dictionary must now include Text as a Data Item.

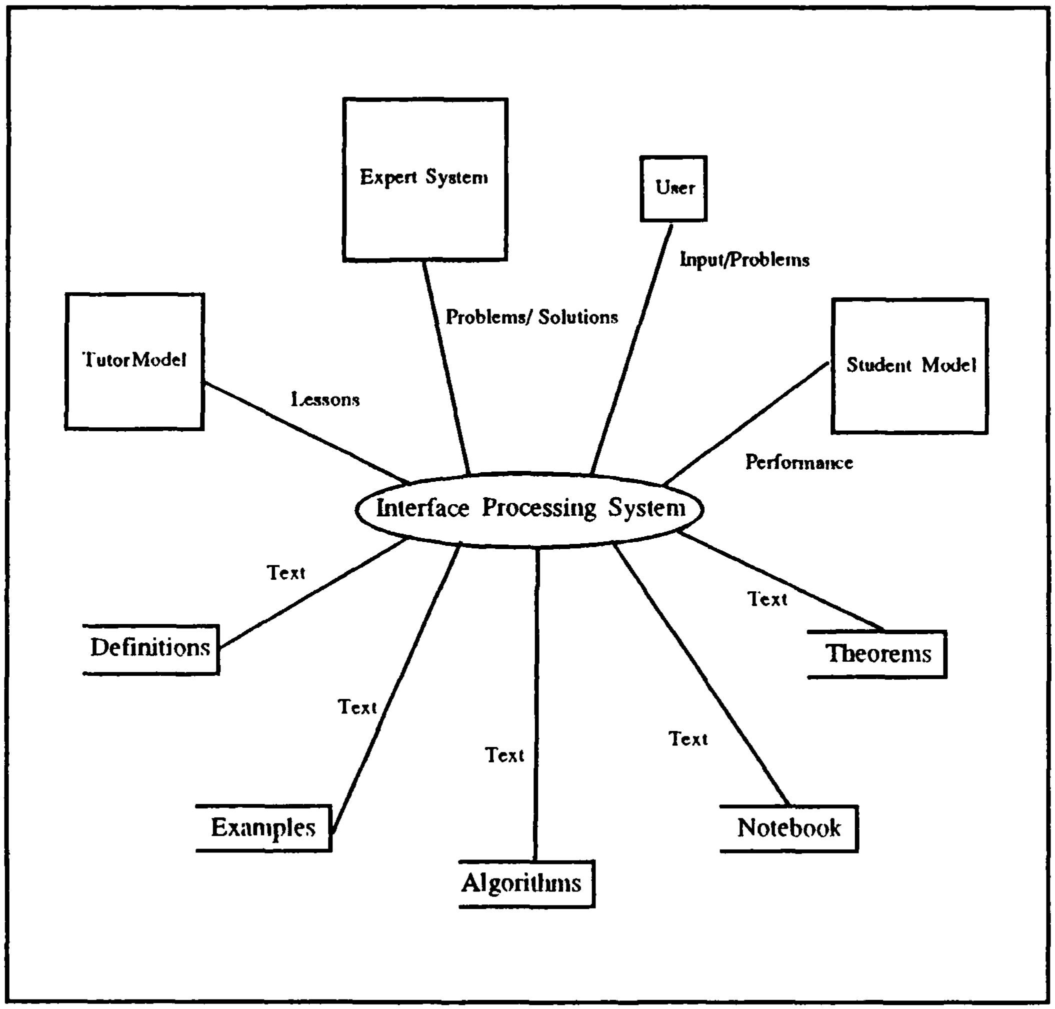


Figure 47 -DFD for the User Interface Process

Still, this representation is too general. Figure 48 shows the DFD for the *Interface Processing System* Process.

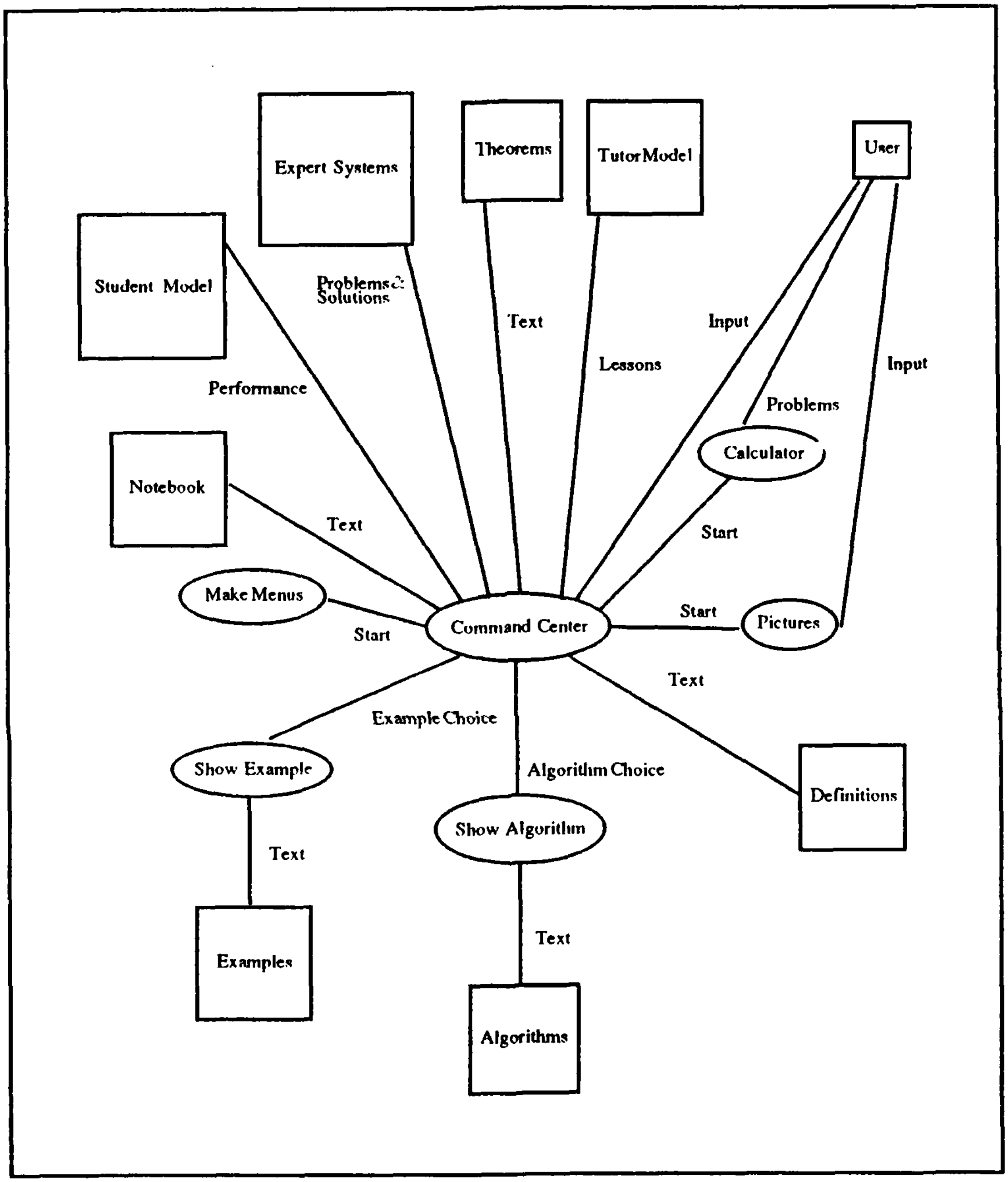


Figure 48 -DFD for the Interface Processing System Process

Observe that all the data stores and processes mentioned in the previous examples are shown as outside entities in this DFD shown in Figure 48. Further notice that all the processes listed in the DFD shown in Figure 48 represent the main functionality of the user interface.

The command center controls all the operations in the *Interface Processing System*. Based on input from the *User*, the Command Center contacts the other entities in the DMT. For example, the user may wish to see a Theorem, Definition or a Notebook entry. The Command Center retrieves that information from the appropriate data store and presents it to the user.

The Calculator process and the Pictures process are programs within the program. This means that once the Command Center starts these processes, they perform their functions based solely on the *User* input and not from any commands from the Command Center. Once the user terminates these two processes, the program passes control back to the Command Center.

The Calculator process performs simple calculations pertaining to any Discrete Math lesson and the Pictures process allows the user to represent Discrete Math problems in a graphical form.

The Show Example and Show Algorithm processes step the user through desired problem examples and Discrete Math algorithms based on the user's choice. These processes are different from simply showing the user a definition or a theorem in that the program presents each example and algorithm like a lesson.

Finally, the Make Menus process constructs the menuing system that allows the *User* to make his choices from the Command Center.

Now the Data Dictionary contains three additional items: Example Choice, Algorithm Choice and Start. The Example and Algorithm Choice options are self-explanatory; but, Start needs some explanation.

When a process receives the Start Data Item, it begins its process. The process executes based solely on internal data structures. It does not rely on any outside Data Stores mentioned in the higher levels of abstraction. As mentioned for the Calculator and Pictures processes, when the running process is complete, it returns control back to the Command Center.

2. The Data Dictionary

From the above Data Flow Diagrams, the Data Dictionary contains the Data Items listed in Figure 49:

Problems
Solutions
Lessons
Performance
Text
Input
Example Choice
Start
Algorithm Choice

Figure 49 - Data Dictionary Entries

The designation assigneu to each data item is a high level abstraction of what is actually represented in the program. More detail is provided by subdividing each data item into its atomic levels. For example, the "Lessons" Data Dictionary entry may divide into two parts: "Text" and "Pictures".

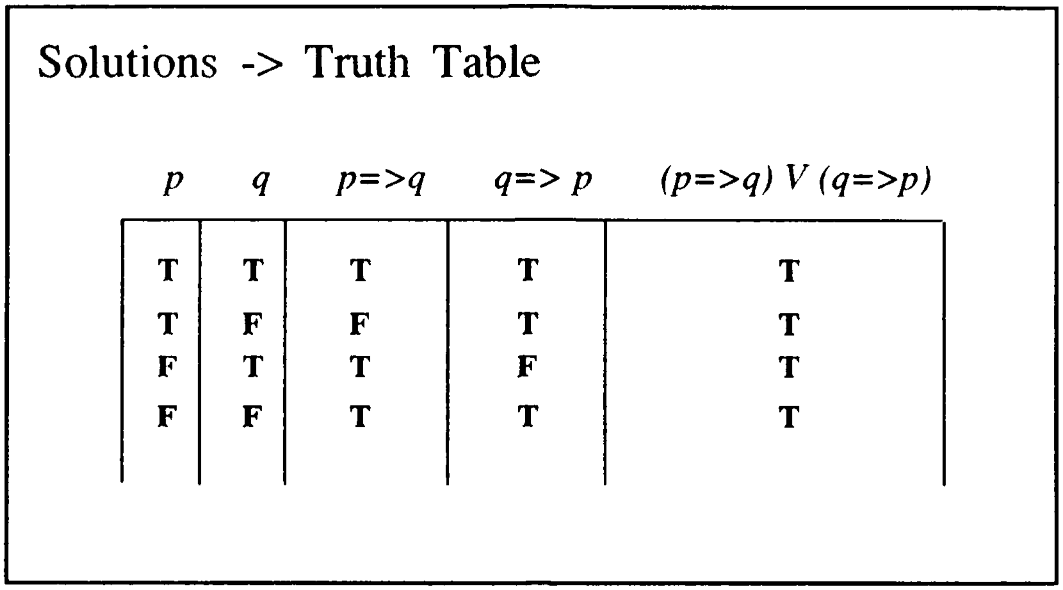
The following description of the Data Dictionary for the DMT is based on one lesson: Logic. Obviously, the DMT will contain more sub-divisions of Data Items as more lessons are added.

The following figures show each entry in the Data Dictionary subdivided into its atomic level. For example, Figure 50 shows that the "Problems" data item is subdivided into a "Logic Equation". Figure 50 also shows what a "Logic Equation" looks like. Figure 51 shows how the "Solutions" data item is subdivided. Figures 52, 53 and 54 show how the "Lessons" data item is subdivided. Figures 55 and 56 show how the "Performance" data item is subdivided. Figures 57, 58, 59, 60 and 61 show how the "Text" data item is subdivided. Figure 62 shows how the "Input", "Example Choice" and "Algorithm Choice" data items are subdivided. Figure 63 shows how the "Start" data item is subdivided.

Problems -> Logic Equation

(p=>q) V (q=>p)

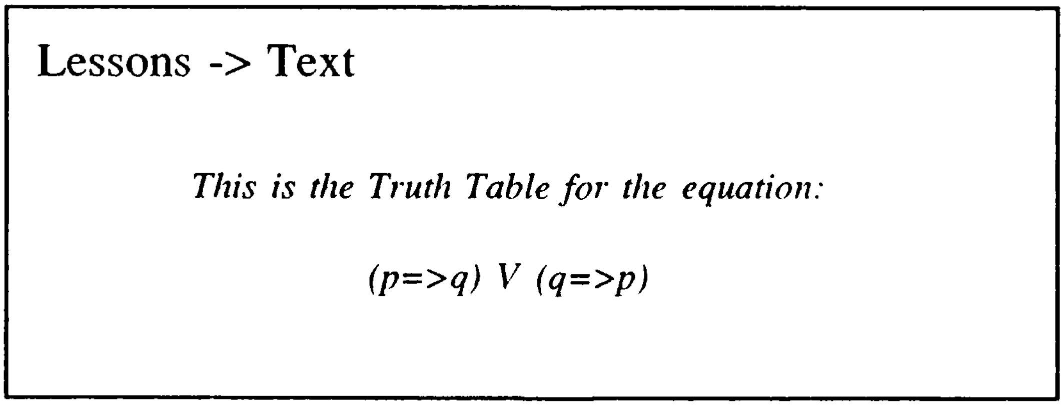
Figure 50 -Problems Data Item



Solutions -> Truth Table

p	*q*	*p=>q*	*q=> p*	*(p=>q) V (q=>p)*
T	**T**	**T**	**T**	**T**
T	**F**	**F**	**T**	**T**
F	**T**	**T**	**F**	**T**
F	**F**	**T**	**T**	**T**

Figure 51 -Solutions Data Item

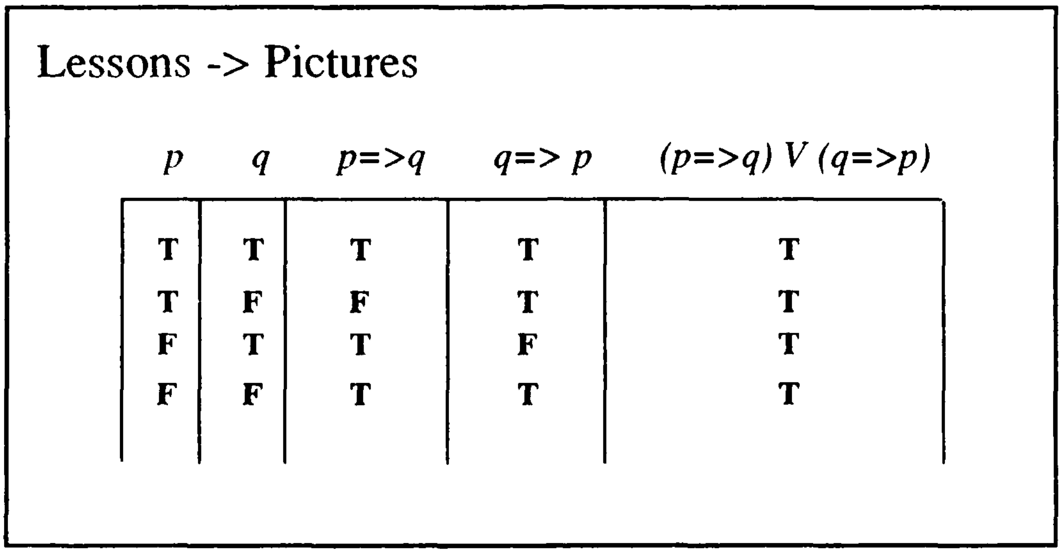


Lessons -> Text

This is the Truth Table for the equation:

(p=>q) V (q=>p)

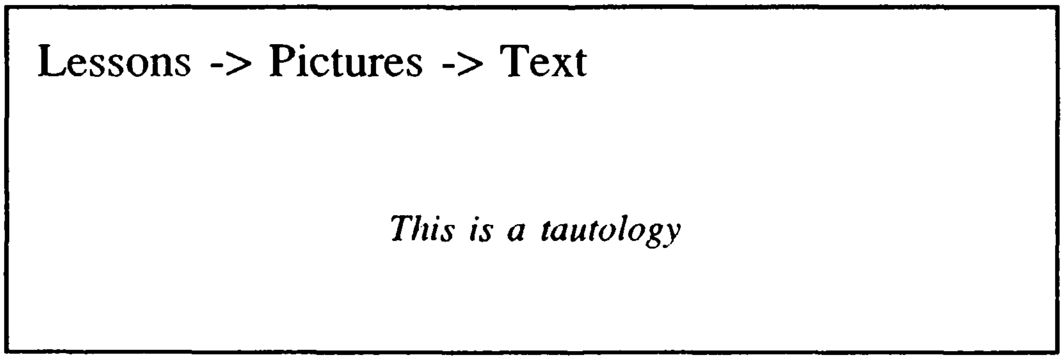
Figure 52 - Lessons Data Item (Text)



Lessons -> Pictures

p	*q*	*p=>q*	*q=> p*	*(p=>q) V (q=>p)*
T	**T**	**T**	**T**	**T**
T	**F**	**F**	**T**	**T**
F	**T**	**T**	**F**	**T**
F	**F**	**T**	**T**	**T**

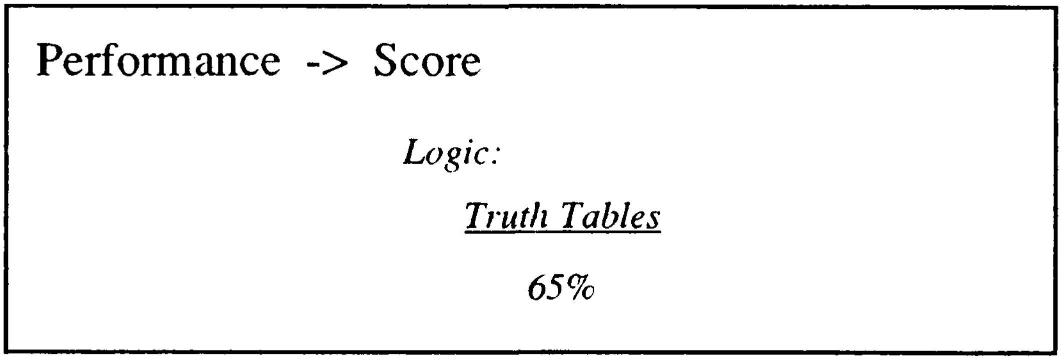
Figure 53 - Lessons Data Item (Pictures)



Lessons -> Pictures -> Text

This is a tautology

Figure 54 - Lessons Data Item (Pictures -> Text)



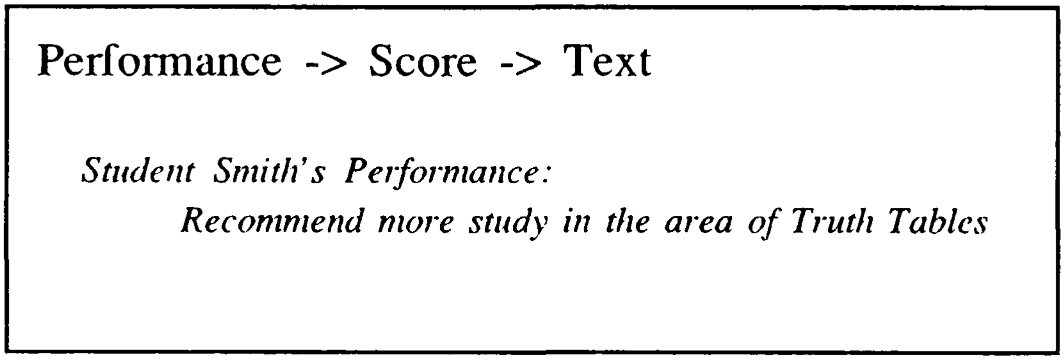
Performance -> Score

Logic:

Truth Tables

65%

Figure 55 - Performance Data Item (Score)

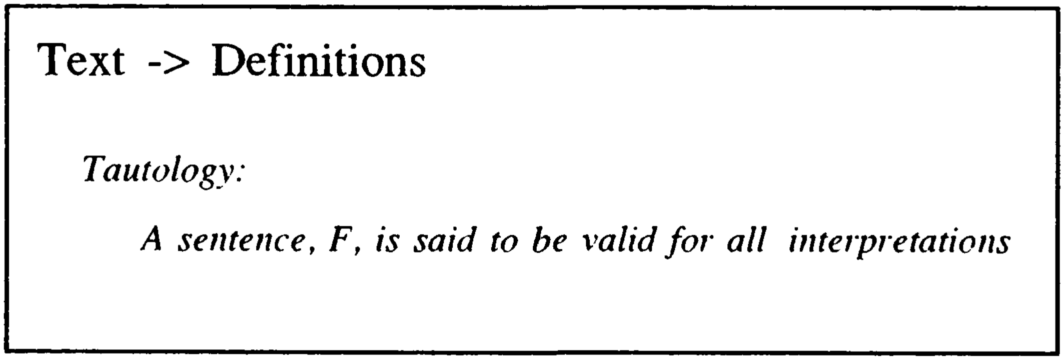


Performance -> Score -> Text

Student Smith's Performance:

Recommend more study in the area of Truth Tables

Figure 56 - Performance Data Item (Score -> Text)

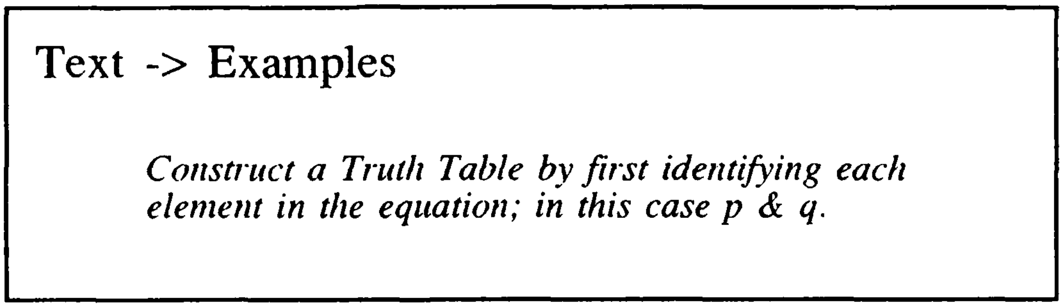


Text -> Definitions

Tautology:

A sentence, F, is said to be valid for all interpretations

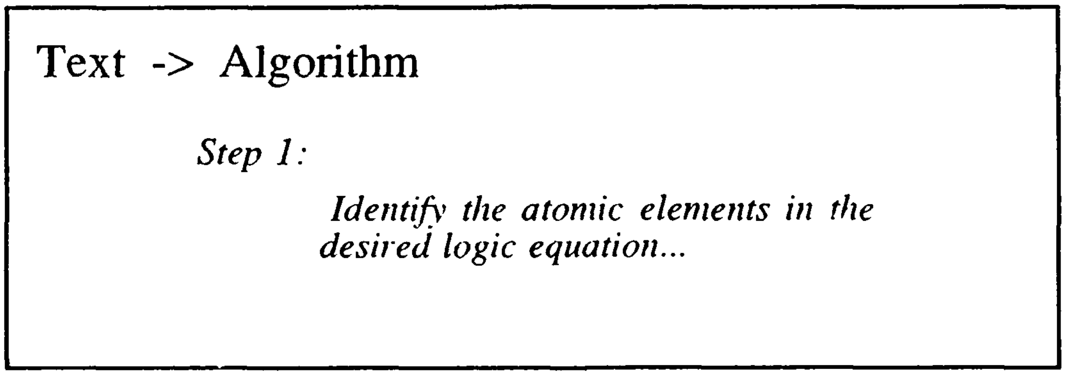
Figure 57 - Text Data Item (Definitions)



Text -> Examples

Construct a Truth Table by first identifying each element in the equation; in this case p & q.

Figure 58 - Text Data Item (Examples)

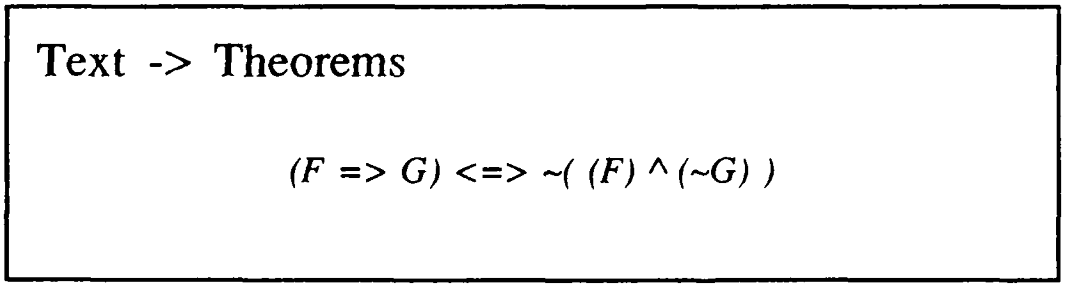


Text -> Algorithm

Step 1:

Identify the atomic elements in the desired logic equation...

Figure 59 - Text Data Item (Algorithm)



Text -> Theorems

(F => G) <=> ~((F) ^ (~G))

Figure 60 - Text Data Item (Theorems)

Text -> Notebook

Any Text Data Item from above

Figure 61 - Text Data item (Notebook)

Input/Example Choice/Algorithm Choice -> Menu Selection

See the Requirements Document

Figure 62 - Input/Example Choice/ Algorithm Choice (Menu Selection)

Start ->

A signal to begin

Figure 63 - Start Data Item

APPENDIX C

HOT KEY SUMMARY

A. BEGIN

1. *Begin* is invoked by typing the letter B.

2. The *Begin* menu provides two options begin a lesson, *Start a lesson* from the first page in the lesson, or *Return to the last active page* of a previous lesson.

B. ESCAPE

1. Escape is invoked by typing the *Esc* key.

2. *Esc* key is accessible throughout the tutor to back out of menus.

C. EXAMS

1. *Exams* is selected from the *main menu* by typing the hot key, E.

2. A particular exam may be chosen by moving the cursor to the name of the exam in the list and pressing the enter key.

D. HELP

1. *Help* is invoked by typing the letter H.

2. With the exception of the *Esc* key, the operation of all hot keys is suspended while *help* is active.

3. The help screen describes the hot keys.

4. Typing the *Esc* key will exit the *help* screen.

5. Typing the *Esc* key twice while the help screen is active will exit the Tutor and return the user to the operating system.

E. INFORMATION

1. *Information* is invoked from the *main menu* by typing the letter I.

2. *Definitions, examples, theorems,* and *proofs* are selectable from the *information* menu.

3. The selected material may be *added to the user's notebook* or may be *directed to the printer* for hard copy.

F. NOTEBOOK

1. *Notebook* is selected from the *main menu* by typing the letter N.

2. The *notebook* may be *displayed* or *printed.*

G. TOOLS

1. *Tools,* is selected from the *main menu* by typing the hot key T.

2. *Diagrams, Reference, Calculator,* or *Problem Solver* are available in the *tools* menu.

H. QUIT

1. *Quit* is selected from the main menu by typing the letter Q.

2. *Save the current position* and *exit* may be selected from the *quit* menu

APPENDIX D

INSTALLATION PROCEDURES

The following files must reside in the same directory for the DMT to execute properly:

GOTH.CHR
LITT.CHR
SANS.CHR
TRID.CHR
EXPL.TXT
EXPL1.TXT
LENGTH.TXT
LOGIC.TXT
LOGICORG.TXT
LOGICSET.TXT
ATT.BGI
CGA.BGI
EGAVGA.BGI
HERC.BGI
IBM8514.BGI
PC3270.BGI
GRAPH.DEF
NOTHING.DEF
DMT.HLP
CALC.EXE
FREE.EXE

DMT.EXE

LSN.EXE

FLASH.EXE

PRINT.EXE

EXAM.EXE

RULES.EXE

TABLE.EXE

TXTMOD.EXE

VENN.EXE

VENNINFO.EXE

APPENDIX E

THE CODE: FILE "DMT.C"

```
/**********************************************************************
                  The Discrete Math Tutor (DMT)
          Thesis Project at the Naval Postgraduate School
           1989-1990 by Keith Calcote and Rick Howard

  LIBRARY CALLS:
            atoi                  Turbo C Lib
            chgonkey              CXL Lib
            error_exit            DMT Utilities
            eror_open_file        DMT Utilities
            exit                  DMT Utilities
            findfirst             Turbo C Lib
            introduction_bar      DMT Utilities
            interface_bar         DMT Utilities
            hidecur               CXL Lib
            normal_exit           DMT Utilities
            setkbloop             CXL Lib
            setonkey              CXL Lib
            set_video             DMT Utilities
            spawnl                Turbo C Lib
            strbtrim              CXL Lib
            top_bar               DMT Utilities
            videoinit             CXL Lib
            waitkey               CXL Lib
            wcenters              CXL Lib
            wclose                CXL Lib
            wfillch               CXL Lib
            wgetchf               CXL Lib
            whelpcat              CXL Lib
            whelpdef              CXL Lib
            whelpopc              CXL Lib
            whelpushc             CXL Lib
            winpbeg               CXL Lib
            winpdef               CXL Lib
            winpread              CXL Lib
            winputsf              CXL Lib
```

LIBRARY CALLS (CONTINUED):

wmenubeg	CXL Lib
wmenuend	CXL Lib
wmenuget	CXL Lib
wmenuitem	CXL Lib
wmenuinext	CXL Lib
wopen	CXL Lib
wpickstr	CXL Lib
wprintf	CXL Lib
wprints	CXL Lib
wputs	CXL Lib
wrjusts	CXL Lib
wtextattr	CXL Lib
wtitle	CXL Lib

PROGRAM CALLS:

calc.exe
exam.exe
flash.exe
lsn.exe
print.exe
rules.exe
table.exe
textmod.exe
venn.exe
venninfo.exe

DMT FUNCTIONS:

main
initialize
calculator
flash_cards
exams
line_inp_demo
logic_exam
main_menu
menudemo
notebook
open_back_wind
open_titl_wind
parse_cmd_line
pick_algorithm
pre_menu1

```
DMT FUNCTIONS (CONTINUED):
                print_notebook
                quit_menu
                rules
                set_video
                table
                tools
                venn
                venninfo
                view_notebook

 COMPLETED: 4/12/90

 PERSONS: Keith Calcote & Rick Howard

 PURPOSE: To develop the user interface module of a Dicrete Math Intelligent
           tutoring system that runs on a IBM PC or compatable.

**********************************************************************/

/* header files */

#include <d:\tc\include\conio.h>
#include <d:\tc\include\ctype.h>
#include <d:\tc\include\dos.h>
#include <d:\tc\include\stdio.h>
#include <d:\tc\include\stdlib.h>
#include <d:\tc\include\string.h>
#include <d:\tc\include\process.h>
#include <d:\tc\include\dir.h>
#include <d:\tc\include\alloc.h>
#include "d:\cxl\cxldef.h"
#include "d:\cxl\cxlkey.h"
#include "d:\cxl\cxlmou.h"
#include "d:\cxl\cxlstr.h"
#include "d:\cxl\cxlvid.h"
#include "d:\cxl\cxlwin.h"
#include "d:\tc\thesis\globals.h"
#include "d:\tc\thesis\defs.h"
#include "d:\tc\thesis\help.h"
#include "d:\tc\thesis\util.h"
#include "d:\tc\thesis\link.c"
#include "d:\tc\thesis\video.h"
```

```
/* function prototypes */

static void initialize (void);
static void calculator(void);
static void flash_cards(void);
static void exams(void);
static void line_inp_demo (void);
static void logic_exam(void);
static void main_menu (void);
static void menudemo(void);
static void notebook(void);
static void open_back_wind(void);
static void open_titl_wind(void);
static void parse_cmd_line(int argc,char *argv[], int *start_up);
static void pick_algorithm(void);
static void pre_menu1 (void);
static void print_notebook(void);
static void quit_menu(void);
static void rules(void);
static void set_video(void);
static void table(void);
static void tools(void);
static void venn(void);
static void venninfo(void);
static void view_notebook(void);
```

```
/**********************************************************************

 FUNCTION  : main

 CALLED BY: NONE

 CALLS       : See Declarations

 MODIFIED  : 4/12/90

 PERSON      : Rick Howard

 PURPOSE    : See Declarations

**********************************************************************/

void main(int argc,char *argv[])
{
  /*
    Initialize the CXL video system, define hot keys and
    define the system's help screen attributes.
  */
  initialize();

  /*
    Process the command line arguments.
  */
  parse_cmd_line(argc,argv,&start_up);

  /*
    If this is the initialial start of the program,
    display the title screen.
  */
  if (start_up){
    open_back_wind();       /* Background to Title Screen */
    set_video();            /* Check for mono, CGA or EGA screen */
    open_titl_wind();       /* Display the title */
    introduction_bar();     /* Display help bar  */
    main_menu();            /* Display the main menu */
  }
  normal_exit(); /* Terminate the program */
}
```

```
/*****************************************************************

 FUNCTION  : initialize

 CALLED BY: dmt

 CALLS        : videoinit
                  setonkey
                  whelpdef

 MODIFIED  : 4/12/90

 PERSON      : Rick Howard
 PURPOSE   : Initializes CXL's video system, defines all hot keys and the
                  the attributes for all help screens.

*****************************************************************/

static void initialize(void)
{
  /*
     Initialize the CXL video system.
  */
  videoinit();

  /*
     Define all hot keys.
  */
  setonkey(0x3062,begin_lsn,0);            /* B */
  setonkey(0x1265,exams,0);                /* E */
  setonkey(0x1769,information,0);          /* I */
  setonkey(0x1474,tools,0);                /* T */
  setonkey(0x316E,notebook,0);             /* N */
  setonkey(0x1071,quit_menu,0);            /* Q */
  setonkey(0x2D78,confirm_quit,0);         /* X */
  setonkey(0x326D,memory,0);               /* M */

  /*
     Define the help screen attributes.
  */
  whelpdef("DMT.HLP",0x2368,BLACK|_LGREY,BLACK|_LGREY,
                                  LBLUE|_LGREY,LRED|_LGREY,pre_help);
}
```

```
/************************************************************************

 FUNCTION  : calculator

 CALLED BY: tools

 CALLS       : spawnl

 MODIFIED  : 4/12/90

 PERSON      : Rick Howard

 PURPOSE    : Suspends the DMT interface program and calls the calc.exe
                     program

************************************************************************/

static void calculator(void)
{
   spawnl(P_WAIT,"calc.exe","calc.exe",NULL);

   /*
      Returns the user to the interface if he is not inside a lesson;
      otherwise, returns the user to the lesson.
   */
   if (!from_lsn)
      menudemo();
   else
      exit(0);
}
```

```
/******************************************************************

 FUNCTION  : flash_cards

 CALLED BY: tools

 CALLS       : spawnl

 MODIFIED  : 4/12/90

 PERSON     : Rick Howard

 PURPOSE    : Suspends the DMT interface program and calls the flash.exe
                      program

/******************************************************************

static void flash_cards(void)
{
  spawnl(P_WAIT,"flash.exe","flash.exe",NULL);

  /*
    Returns the user to the interface if he is not inside a lesson;
    otherwise, returns the user to the lesson.
  */
  if (!from_lsn)
    menudemo();
  else
    exit(0);

}
```

```
/*********************************************************************

 FUNCTION  : exams

 CALLED BY: Hot key defined in function initialize

 CALLS       : wmenubeg
               wmenuitem
               wmenuend
               wmenuget
               error_exit

 MODIFIED : 4/12/90

 PERSON     : Rick Howard

 PURPOSE   : Defines the Exam menu and presents the menu to the user

*********************************************************************/

static void exams(void)
{
  int selection;

  wmenubeg(2,31,4,42,0,YELLOW|_BLUE,YELLOW|_BLUE,add_shadow);
  wmenuitem(0,0,"Logic Exam",'L',10,0,logic_exam,0,H_EXAMS);
  wmenuend(10,M_PD|M_SAVE,0,1,YELLOW|_BLUE,LCYAN|_BLUE,0
                                        ,YELLOW|_LGREY);


  selection=wmenuget();
  if(selection==-1&&_winfo.errno>W_ESCPRESS) error_exit(1);
}
```

```
/*********************************************************************

 FUNCTION  : logic_exam

 CALLED BY: exams

 CALLS        : chgonkey
                  wopen
                  error_exit
                  add_shadow
                  wtitle
                  winpbeg
                  wprints
                  windef
                  winpread
                  atoi
                  wputs
                  wgetchf
                  wclose
                  spawnl
                  exit

 MODIFIED  : 4/12/90

 PERSON      : Rick Howard

 PURPOSE   : Allows user to take the logic exam by first asking how many
                  questions he desires.  Then it suspends the DMT program
                  while the Q_&_A.exe program executes.

*********************************************************************/

static void logic_exam(void)
{
   struct _onkey_t *k1;        /*  Linked list of hot keys                    */
   char *num_questions;      /*  # of exam questions the user desires */
   register int response;     /*  Records the user's response           */

   /*
      Assign the current hot key list to k1 and set the current hot
      key list to NULL.
   */
   k1 = chgonkey(NULL);
```

```
/*
  Open a window to retrieve the desired number of exam questions
  from the user.
*/
if(!wopen(10,8,17,70,1,LCYAN|_BLUE, LCYAN|_BLUE)) error_exit(1);
add_shadow();
wtitle("[Enter the Number of Exam Questions Desired]",TLEFT, LCYAN|_BLUE);

/*
  Open window to ask how many exam questions the user desires.
*/
do{
  /*
    Define the window attributes.
  */
  winpbeg(LGREEN|_LGREY,WHITE|_LGREY);

  /*
    Display prompts and define fields.
  */
  wprints( 1, 3, WHITE|_BLUE, "How many exam questions do you wish?");
  winpdef( 1, 41, num_questions, "##",0,0,NULL,0);

  /*
    Mark end of form and process it.
  */
  if(winpread()) break;

  /*
    Verify user's answer.
  */
  if (!wopen(15,24,19,57,0,WHITE|_CYAN,WHITE|_CYAN)) error_exit(1);
  add_shadow();
  wputs("\n Is this information correct? \033A\076Y\b");
  response = wgetchf("YN",’Y’);
  wclose();
}

while (response != ’Y’);
```

```
    /*
       Reset the hot key list.
    */
    chgonkey(k1);
    /*
       Suspend the DMT progiam and launch the Q_&_A.exe program.
    */
    spawnl(P_WAIT."Q_&_A.exe","Q_&_A.exe",
               num_questions,"test1.txt","expl1.txt",NULL);

    /*
       Returns the user to the interface if he is not inside a lesson;
       otherwise, returns the user to the lesson.
    */
    if (!from_lsn)
       menudemo();
    else
       exit(0);
    wclose();
 }
```

```
/*********************************************************************

 FUNCTION  : main_menu

 CALLED BY: dmt

 CALLS      : whelpushc
              wmenubeg
              wmenuitem
              wmenuend
              wmwnuget
              whelpopc

 MODIFIED  : 4/12/90

 PERSON     : Rick Howard

 PURPOSE    : Defines and executes the main menu on the title screen.

*********************************************************************/

static void main_menu(void)
{
   /*
      Push the initial help screen onto the help screen stack.
   */
   whelpushc(H_INITIAL);

   /*
      Define and process the main menu.
   */
   wmenubeg(13,27,16,53,0,LBLUE|_BLUE,LBLUE|_BLUE,pre_menu1);
   wmenuitem(0,0,"Start Demo",'S',1,M_CLOSB,menudemo,0,0);
   wmenuitem(1,0,"Exit demo" ,'E',6,0,NULL ,0,0);
   wmenuend(1,M_VERT,25,3,LCYAN|_BLUE,WHITE|_BLUE,0,BLUE|_LGREY);

   if(wmenuget()==-1) if(_winfo.errno>W_ESCPRESS) error_exit(1);

   /* pop the global help category off of the stack, and into the void */
   whelpopc();
}
```

```
/*************************************************************************

 FUNCTION  : menudemo

 CALLED BY: calculator
            flash_cards
            logic_exam
            main_menu
            pickalgorithm
            table
            rules
            venn
            venninfo
            view_notebook

 CALLS     : whelpushc
            wopen
            error_exit
            top_bar
            interface_bar
            waitkey

 MODIFIED : 4/12/90

 PERSON   : Rick Howard

 PURPOSE  : Presents the interface screen and waits for the user to hit
            a hot key.

*************************************************************************/

static void menudemo(void)
{
  /*
    Open the interface window.
  */
  if((w[1]=wopen(2,0,23,79,1,YELLOW|_BLUE,YELLOW|_BLUE))==0)
    error_exit(1);
    /*
       Draw the menu bar and the help bar.
    */
    top_bar();
    interface_bar();

    /*
       Push the user interface help scrren onto the help stack.
    */
    whelpushc(H_USER_INTERFACE);

    /*
       Wait for the user to choose a hot key.
    */
    while (waitkey() != "!");
}
```

```
/**********************************************************************

 FUNCTION  : notebook

 CALLED BY: parse_cmd_line

 CALLS       : wmenubeg
               wmenuitem
               wmenuend
               whelpcat
               error_exit

 MODIFIED  : 4/12/90

 PERSON     : Rick Howard

 PURPOSE    : Defines the Notebook menu and presents the menu to the user.

**********************************************************************/

static void notebook(void)
{
   int selection;

   wmenubeg(2,56,5,71,0,YELLOW|_BLUE,YELLOW|_BLUE,add_shadow);
   wmenuitem(0,0,"View Notebook",'V',60,0,
                       view_notebook,0,H_VIEW_NOTEBOOK_HELP);
   wmenuitem(1,0,"Print Notebook",'P',61,0,
                      print_notebook,0,H_PRINT_NOTEBOOK);
   wmenuend(60,M_PD|M_SAVE,0,1,YELLOW|_BLUE,LCYAN|_BLUE,0
                                           ,YELLOW|_LGREY);

   selection=wmenuget();
   if(selection==-1&&_winfo.ermo>W_ESCPRESS) error_exit(1);
   whelpcat(H_USER_INTERFACE);
}
```

```
/******************************************************************

 FUNCTION  : open_back_wind

 CALLED BY: dmt

 CALLS        : wopen
                  wprintf
                  error_exit

 MODIFIED  : 4/12/90

 PERSON      : Rick Howard

 PURPOSE    : Draws the background for the title screen

******************************************************************/

static void open_back_wind(void)
{
   register int i;

   if(!wopen(0,0,23,79,5,0,LGREEN|_GREEN)) error_exit(1);
   for(i=1;i<320;i++) wprintf("\033F%cDMT   ",i);
}
```

```
/**********************************************************************

 FUNCTION  : open_titl_wind

 CALLED BY: dmt

 CALLS       : wopen
               error_exit
               add_shadow
               wcenters

 MODIFIED  : 4/12/90

 PERSON      : Rick Howard

 PURPOSE    : Draws the title window for the title screen.

**********************************************************************/

static void open_titl_wind(void)
{
  if(!wopen(1,12,9,67,0,LRED|_MAGENTA,LRED|_MAGENTA)) error_exit(1);
  add_shadow( ):
  wcenters(0,WHITE|_MAGENTA,"Welcome to the Discrete Math Tutor (DMT)
              Prototype!");
  wcenters(2,LCYAN|_MAGENTA,"DMT: 1989-1990");
  wcenters(3,LCYAN|_MAGENTA,"by");
  wcenters(4,LCYAN|_MAGENTA,"Rick Howard");
  wcenters(5,LCYAN|_MAGENTA,"&");
  wcenters(6,LCYAN|_MAGENTA,"Keith Calcote");
}
```

```
/**********************************************************************

 FUNCTION  : parse_cmd_line

 CALLED BY: dmt

 CALLS       : tools
               quit_menu
               notebook

 MODIFIED  : 4/12/90

 PERSON      : Rick Howard

 PURPOSE     : Allows the user to call specific funtions residing in the DMT
                program and exit while the lsn program is running.  For
                example, a user taking the logic lesson can invoke the tools
                function in the DMT.

**********************************************************************/

static void parse_cmd_line(int argc,char *argv[],int *start_up)
{
  char *p;   /* The character that represents the cmd line argument */

  /*
    If there exists command line arguments...
  */
  if(argc > 1)
  {
    p=argv[1];
    if ((*p == ’T’) || (*p == ’t’))
    {
      from_lsn = TRUE;
      tools();
      from_lsn = FALSE;
      *start_up = 0;
    }
    else if (*p == ’Q’)
    {
      quit_menu();
      *start_up = 0;
    }
```

```
        else if (*p == 'N')
        {
          from_lsn = TRUE;
          notebook();
          from_lsn = FALSE;
          *start_up = 0;
        }
      }
    }
```

```
/************************************************************************

 FUNCTION  : pre_menu1

 CALLED BY: main_menu

 CALLS        : hidecur
                 add_shadow

 MODIFIED  : 4/12/90

 PERSON      : Rick Howard

 PURPOSE    : Hides the cursor and adds a shadow to a menu.

************************************************************************/

static void pre_menu1(void)
{
  hidecur();
  add_shadow();
}
```

```
/********************************************************************

 FUNCTION  : print_notebook

 CALLED BY: notebook

 CALLS       : chgonkey
               wopen
               error_exit
               add_shadow
               wtitle
               winpbeg
               wprints
               winpdef
               winpread
               wputs
               wgetchf
               findfirst
               spawnl
               error_open_file
               wclose

 MODIFIED  : 4/12/90

 PERSON     : Rick Howard

 PURPOSE    : Queries the user for his personalized notebook name and
               sends that file to the program print.exe.

********************************************************************/

static void print_notebook(void)
{
   int done;                   /* Used to indicate if the notebook
                                  file can be found                  */

   struct ffblk ffblk;         /* Space filler used in the
                                  findfirst function                 */

   struct _onkey_t *k1;  /* Points to the current hot-key list      */

   register int response; /* Accepts user's response                */
```

```
/*
   Set k1 = to the current hot-key list and disable all
   hot-key definitions.
*/
k1 = chgonkey(NULL);

/*
   Open the window.
*/
if(!wopen(10,8,17,70,1,LCYAN|_BLUE, LCYAN|_BLUE)) error_exit(1);
add_shadow();
wtitle("[Name Your Personalized Notebook]",TLEFT, LCYAN|_BLUE);

/* Display prompts and define fields. */
do{
   winpbeg(LGREEN|_LGREY,WHITE|_LGREY);
   wprints( 1, 3, WHITE|_BLUE, "What is your Notebook Name?");
   winpdef( 1, 35, notebook_name, "WWWWWWWWWWWW",0,0,NULL,0);

   /*
      Mark end of form and process it.
   */
   if(winpread()) break;

   /*
      Ensure that the user information is correct.
   */
   if (!wopen(15,24,19,57,0,WHITE|_CYAN,WHITE|_CYAN)) error_exit(1);
   add_shadow();
   wputs("\n Is this information correct? \033A\076Y\b");
   response = wgetchf("YN",'Y');
   wclose();
}
while (response != 'Y');

/*
   Find the user's notebook in the current directory.
*/
done = findfirst(notebook_name, &ffblk, 0);
```

```
  /*
     If the user's notebook is found in the current directory,
     send the user's notebook name to the program print.exe.
     Otherwise, display an error message.
  */
  if (done == 0){
     spawnl(P_WAIT, "print.exe", "print.exe", notebook_name, NULL);
  }
  else
     error_open_file(notebook_name);

  /*
     Close the window and enable the hot-key list again.
  */
  wclose();
  chgonkey(k1);

}
```

```
/*********************************************************************

 FUNCTION  : quit_menu

 CALLED BY: initialize
             parse_cmd_line

 CALLS      : wmenubeg
              wmenuitem
              wmenuend
              wmenuget
              error_exit
              whelpcat

 MODIFIED  : 4/12/90

 PERSON     : Rick Howard

 PURPOSE    : Displays the quit menu to the user

*********************************************************************/

static void quit_menu(void)
{
  int selection;  /*  The user's menu selection  */

  /*
    Define the menu structure.
  */
  wmenubeg(2,55,5,77,0,YELLOW|_BLUE,YELLOW|_BLUE,add_shadow);
  wmenuitem(0,0,"Save Current Position",'!',70,0,
                    do_nothing,0,H_UNAVAILABLE);
  wmenuitem(1,0,"Exit",'E',71,M_CLOSE,confirm_quit,0,H_EXIT);
  wmenuend(70,M_PD|M_SAVE,0,1,YELLOW|_BLUE,LCYAN|_BLUE,0
                                    ,YELLOW|_LGREY);

  /*
    Process the menu
  */
  selection=wmenuget();
  if(selection==-1&&_winfo.errno>W_ESCPRESS) error_exit(1);
  whelpcat(H_USER_INTERFACE);
}
```

```
/**********************************************************************

 FUNCTION  : table

 CALLED BY: tools

 CALLS        : spawnl
                menudemo
                exit

 MODIFIED  : 4/12/90

 PERSON       : Rick Howard

 PURPOSE      : Suspend the dmt.exe program and launch the table.exe program.

**********************************************************************/

static void table(void)
{
  spawnl(P_WAIT,"table.exe","table.exe",NULL);

  /*
    Returns the user to the interface if he is not inside a lesson;
    otherwise, returns the user to the lesson.
  */
  if (!from_lsn)
    menudemo( );
  else
    exit(0);
}
```

```
/**********************************************************************

 FUNCTION  : tools

 CALLED BY: initialize
            parse_cmd_line

 CALLS      : wmenubeg
              wmenuitem
              wmenuend
              wmenuget
              error_exit
              whelpcat

 MODIFIED  : 4/12/90

 PERSON     : Rick Howard

 PURPOSE    : Displays the tools menu to the user.

**********************************************************************/

static void tools(void)
{
  int selection;  /*  The user's menu selection   */

  /*
    Define the menu structure.
  */
  wmenubeg(2,42,7,57,0,YELLOW|_BLUE,YELLOW|_BLUE,add_shadow);
  wmenuitem(0,0,"Diagrams",'D',10,0,NULL,0,H_PICTURES);

    wmenubeg(8,42,10,56,0,YELLOW|_BLUE,YELLOW|_BLUE,add_shadow);
    wmenuitem(0,0,"Venn Diagrams",'V',51,0,venn,0,H_VENN_DIAGRAM_PICS);
    wmenuend(51,M_PD|M_SAVE,0,1,YELLOW|_BLUE,LCYAN|_BLUE,0
                                          ,YELLOW|_LGREY);

  wmenuitem(1,0,"Reference",'R',11,0,NULL,0,H_REFERENCE);

    wmenubeg(8,42,11,55,0,YELLOW|_BLUE,YELLOW|_BLUE,add_shadow);
    wmenuitem(0,0,"Truth Tables",'T',70,0,
                     do_nothing,0,H_TRUTH_TABLE_REF);
```

```
      wmenubeg(10,42,13,48,0,YELLOW|_BLUE,YELLOW|_BLUE,add_shadow);
      wmenuitem(0,0,"Drill",'D',80,0,flash_cards,0,H_TRUTH_TABLE_DRILL);
      wmenuitem(1,0,"Rules",'R',81,0,rules,0,H_TRUTH_TABLE_RULES);
      wmenuend(80,M_PD|M_SAVE,0,1,YELLOW|_BLUE,LCYAN|_BLUE,0
                                    ,YELLOW|_LGREY);

    wmenuitem(1,0,"Venn Diagrams",'V',71,0,
                    venninfo,0,H_TRUTH_TABLE_REF);
    wmenuend(70,M_PD|M_SAVE,0,1,YELLOW|_BLUE,LCYAN|_BLUE,0
                                    ,YELLOW|_LGREY);

  wmenuitem(2,0,"Calculator",'C',12,0,calculator,0,H_CALCULATOR);
  wmenuitem(3,0,"Problem Solver",'P',13,0,
                  do_nothing,0,H_PROBLEM_SOLVER);

    wmenubeg(8,42,10,55,0,YELLOW|_BLUE,YELLOW|_BLUE,add_shadow);
    wmenuitem(0,0,"Truth Tables",'T',40,0,
                    table,0,H_TRUTH_TABLE_PROBLEM_SOLVER);
    wmenuend(40,M_PD|M_SAVE,0,1,YELLOW|BLUE,LCYAN|_BLUE,0
                                    ,YELLOW|_LGREY);

  wmenuend(10,M_PD|M_SAVE,0,1,YELLOW|_BLUE,LCYAN|_BLUE,0
                                    ,YELLOW|_LGREY);
  /*
    Process the menu.
  */
  selection=wmenuget();
  if(selection==-1&&_winfo.errno>W_ESCPRESS) error_exit(1);
  whelpcat(H_USER_INTERFACE);
}
```

```
/**********************************************************************

 FUNCTION  : rules

 CALLED BY: tools

 CALLS       : spawnl
               menudemo
               exit

 MODIFIED  : 4/12/90

 PERSON      : Rick Howard

 PURPOSE    : Suspend the dmt.exe program and launch the rules.exe program.

**********************************************************************/

static void rules(void)
{
  spawnl(P_WAIT,"rules.exe","rules.exe",NULL);

  /*
    Returns the user to the interface if he is not inside a lesson;
    otherwise, returns the user to the lesson.
  */
  if (!from_lsn)
    menudemo( );
  else
    exit(0);
}
```

```
/*********************************************************************

 FUNCTION  : venn

 CALLED BY: tools

 CALLS       : spawnl
                 menudemo
                 exit

 MODIFIED  : 4/12/90

 PERSON      : Rick Howard

 PURPOSE    : Suspend the dmt exe program and launch the venn.exe program.

*********************************************************************/

static void venn(void)
{
  spawnl(P_WAIT,"venn.exe","venn.exe",NULL);

  /*
    Returns the user to the interface if he is not inside a lesson;
    otherwise, returns the user to the lesson.
  */
  if (!from_lsn)
    menudemo();
  else
    exit(0);

}
```

```
/************************************************************************

 FUNCTION  : venninfo

 CALLED BY: tools

 CALLS      : spawnl
              menudemo
              exit

 MODIFIED  : 4/12/90

 PERSON     : Rick Howard

 PURPOSE    : Suspend the dmt.exe program and launch the venninfo.exe
              program.

************************************************************************/

static void venninfo(void)
{
  spawnl(P_WAIT,"venninfo.exe","venninfo.exe",NULL);

  /*
    Returns the user to the interface if he is not inside a lesson;
    otherwise, returns the user to the lesson.
  */
  if (!from_lsn)
    menudemo();
  else
    exit(0);

}
```

```
/*********************************************************************

 FUNCTION  : view_notebook

 CALLED BY: notebook

 CALLS        : chgonkey
                 wopen
                 error_exit
                 add_shadow
                 wtitle
                 winpbeg
                 wprints
                 winpdef
                 winpread
                 wputs
                 wgetchf
                 wclose
                 findfirst
                  strbtrim
                 spawnl
                 menudemo
                  exit
                 error_open_file

 MODIFIED  : 4/12/90

 PERSON      : Rick Howard

 PURPOSE    : Prompts the user for his personalized notebook name and sends
                 that name to the lsn.exe program.

*********************************************************************/

static void view_notebook(void)
{
  struct ffblk ffblk;            /*  Used as a place filler in the
                                     findfirst function                    */

  struct _onkey_t *k1;    /*  Points to the defined hot-key list             */
```

```
int done;              /* Used to indicate if the user's
                          personalized notebook name is found in
                          the current directory                  */

register int response; /* Holds the user's response              */

/*
   k1 points to the current hot-key list and all hot-keys are
   disabled.
*/
k1 = chgonkey(NULL);

/*
   Open the window.
*/
if(!wopen(10,8,17,70,1,LCYAN|_BLUE, LCYAN|_BLUE)) error_exit(1);
add_shadow();
wtitle("[Name Your Personalized Notebook]",TLEFT, LCYAN|_BLUE);

/*
   Display prompts and define fields.
*/
do{
   winpbeg(LGREEN|_LGREY,WHITE|_LGREY);

   wprints( 1, 3, WHITE|_BLUE, "What is your Notebook Name?");
   winpdef( 1, 35, notebook_name, "WWWWWWWWWWWWW",0,0,NULL,0);

   /*
      Mark end of form and process it.
   */
   if(winpread()) break;
```

```
    /*
       Ensure that the user information is correct.
    */
    if (!wopen(15,24,19,57,0,WHITE|_CYAN,WHITE|_CYAN)) error_exit(1);
    add_shadow();
    wputs("\n Is this information correct? \033A\076Y\b");
    response = wgetchf("YN",'Y');
    wclose();
}

while (response != 'Y');

/*
    Enable the hot-key list.
*/
chgonkey(k1);

/*
    Find the user's notebook in the current directory.
*/
done = findfirst(notebook_name, &ffblk, 0);

/*
    If the user's notebook is found in the current directory,
    modify it with the program txtmod.exe and send the
    results to the program lsn.exe.  Otherwise, display an
    error message.
*/
if (done == 0)
{
    strbtrim(notebook_name);
    spawnl(P_WAIT,"txtmod.exe","txtmod.exe",notebook_name,
               "notebook.out",NULL);

    spawnl(P_WAIT,"lsn.exe","lsn.exe","notebook.len","notebook.txt",NULL);
```

```
            /*
               Returns the user to the interface if he is not inside a lesson;
               otherwise, returns the user to the lesson.
            */
            if (!from_lsn)
                menudemo();
            else
                exit(0);
        }
        else
            error_open_file(notebook_name);

        /*
            Close the window.
        */
        wclose();

    }
```

APPENDIX F

THE CODE: FILE "LSN.C"

```
/***********************************************************************
                     The Discrete Math Tutor (DMT)
               Thesis Project at the Naval Postgraduate School
                 1989-1990 by Keith Calcote and Rick Howard

LIBRARY CALLS:
          atoi                        Turbo C Lib
          error_exit                  DMT Utilities
          exit                        DMT Utilities
          fclose                      Turbo C Lib
          fopen                       Turbo C Lib
          find_card                   Link Utilitites
          fread                       Turbo C Lib
          fseek                       Turbo C Lib
          getche                      Turbo C Lib
          initiailize_linked_list     Link Utilities
          setonkey                    CXL Lib
          set_video                   DMT Utilities
          spawnl                      Turbo C Lib
          top_bar                     DMT Utilities
          wactiv                      CXL Lib
          waitkey                     CXL Lib
          wborder                     CXL Lib
          wclose                      CXL Lib
          whelpcat                    CXL Lib
          whelpdef                    CXL Lib
          wmenubeg                    CXL Lib
          wmenuend                    CXL Lib
          wmenuget                    CXL Lib
          wmenuitem                   CXL Lib
          wmessage                    CXL Lib
          wopen                       CXL Lib
          wprintf                     CXL Lib
          write_file                  Link Utilitites
```

```
PROGRAM CALLS:
            dmt.exe

LSN FUNCTIONS:
            continue_lsn
            enter_page
            lsn_bar
            main
            notebook
            pageclr
            page_down
            page_up
            quit
            quit_menu
            save_position
            top_bar
            tools

COMPLETED:      4/12/90

PERSONS:        Keith Calcote & Rick Howard

PURPOSE:        To display a lsn inside the DMT user interface.

************************************************************************/
```

```
/* Header Files */

#include <stdio.h>
#include <process.h>
#include <bios.h>
#include <alloc.h>
#include <dir.h>
#include "d:\cxl\cxlstr.h"

#include "d:\cxl\cxlwin.h"
#include "d:\cxl\cxlkey.h"
#include "d:\cxl\cxlvid.h"
#include "d:\tc\thesis\globals.h"
#include "d:\tc\thesis\defs.h"
#include "d:\tc\thesis\help.h"
#include "d:\tc\thesis\util.h"
#include "d:\tc\thesis\link.c"
#include "d:\tc\thesis\video.h"
/*----------------------------------------------------------------*/
```

```
/* function prototypes */

static void add_shadow(void);
static void continue_lsn(void);
static void error_exit(int errnum);
static void enter_page(void);
static void information(void);
static void notebook(void);
static int pageclr(void);
static void page_down(void);
static void page_up(void);
static void quit(void);
static void quit_menu(void);
static void save_position(void);
static void set_video(void);
static void tools(void);
/*----------------------------------------------------------------------*/

/*  Constants  */

#define LEN 50
#define PAGEL 1000
/*-----------------------------------------------------------------------*/
```

```
/* Global Variables */

static WINDOW w[10];        /* Array of window handles   */

static FILE *fptr1,       /*  Pointer to the lesson length file */

static FILE *fptr2 ;      /*  Pointer to the lesson text file   */

static int ch;            /*  Used to get the user's response   */

static int recno ;          /*  Indicates the page number for the lesson */

static int temp,temp1 ;   /*  Used to calculate the user's desired page
                         number.                       */

static char page[PAGEL] ; /*  Holds the contents of the lesson          */

static char *ARGS[3];     /*  Holds the arguments needed to save the
                         user's position in any lesson          */
/*--------------------------------------------------------------------*/
```

```
/**********************************************************************

 FUNCTION :      main
 CALLED BY:      NONE
 CALLS    :      See Declarations
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard & Keith Calcote
 PURPOSE  :      See Declarations

**********************************************************************/

void main(int argc,char *argv[])
{
   /*
      Define all hot keys.
   */
   setonkey(0x5100,page_down,0);         /* Page down */
   setonkey(0x4900,page_ûp,0);           /* Page up */
   setonkey(0x1900,enter_page,0);        /* ALT P */
   setonkey(0x3062,begin_lsn,0);         /* B */
   setonkey(0x1769,information,0);       /* I */
   setonkey(0x1474,tools,0);             /* T */
   setonkey(0x316E,notebook,0);          /* N */
   setonkey(0x1071,quit_menu,0);         /* Q */
   setonkey(0x2D78,quit,0);              /* X */
   setonkey(0x326D,memory,0);            /* M */

   /*
      Define the help screen attributes.
   */
   whelpdef("DMT.HLP",0x2368,BLACK|_LGREY.BLACK|_LGREY.
                                      LBLUE|_LGREY,LRED|_LGREY,pre_help);

   /*
      Draw the menu bar and the help bar.
   */
   top_bar();
   lsn_bar();

   /*
      Check for mono, CGA or EGA screen.
   */
   set_video();
```

```
/*
   Open a window to display the user selected lesson.
*/
if((w[1]=wopen(2,0,23,79,3,WHITE|_CYAN,WHITE|_CYAN))==0)
   error_exit(1);
wmessage("[PgUp/PgDn]",BT_BORD,9,YELLOW|_BLACK);

/*
   Open the Lesson Length file and store the name for use by
   the save_position function.
*/
if( (fptr1=fopen(argv[1],"r")) ==NULL )
{
   wprintf("CAN'T OPEN THIS FILE:
   %s\n",argv[1]);
   waitkey();
   exit(0);
}
else
   ARGS[1] = argv[1];

/*
   Open the Lesson Text file and store the name for use by
   the save_position function.
*/
if( (fptr2=fopen(argv[2],"rb")) ==NULL )
{
   wprintf("CAN'T OPEN THIS FILE:
   %s\n", argv[2]);
   waitkey();
   exit(0);
}
else
   ARGS[2] = argv[2];

/*
   Read the lesson length file.
*/
fread(&length,sizeof(length),1,fptr1);
```

```
    /*
       Determine the user defined page number.
    */
    if(argv[3] != NULL)
       recno = atoi(argv[3]);
    else
       recno = 1;

    /*
       Begin the lesson.
    */
    continue_lsn();
}
```

```
/**********************************************************************

 FUNCTION :        continue_lsn
 CALLED BY:        lsn
                   enter_page
                   notebook
                   page_down
                   page_up
                   tools
 CALLS   :         whelpcat
                   wactiv
                   wmessage
                   wprintf
                   fseek
                   pageclr
                   fread
                   wputns
                   waitkey
                   werrmsg

 MODIFIED :        4/12/90
 PERSON  :         Rick Howard & Keith Calcote
 PURPOSE  :        Displays the proper page number to a lesson based upon
                   the user's input.

**********************************************************************/

static void continue_lsn(void)
{
  /*
    Set the help screen that applies to any generic lesson.
  */
  whelpcat(H_LSN_HELP);

  /*
    Make the lesson window the active window.
  */
  wactiv(w[1]);

  /*
    Display a help message across the border of the lesson window.
  */
  wmessage("[PgUp/PgDn]",BT_BORD,9,YELLOW|_BLACK);
```

```
    /*
       Display the page number.
    */
    wprintf("\r\n\t\t\t\t\t\t\t\tpage %d\r\n",recno) ;

    /*
       Get the current page from disk and display it in the active window.
    */
    offset = length[recno] ;
    if( fseek(fptr2,offset,0) != 0)
    {
       wprintf("CAN'T MOVE POINTER THERE") ;
       exit(0) ;
    }
    pageclr() ;
    fread(page,length[recno+1]-length[recno],1,fptr2) ;
    wputns(page,length[recno+1]-length[recno]);

    /*
       If last page of the lesson, cycle back to the first page.
    */
    if(recno > length[0])
       recno = 1;

    /*
       Wait for the user's response.
    */
    while (waitkey() != 0x4C35);
}
```

```
/************************************************************************

 FUNCTION :      enter_page
 CALLED BY:      lsn
 CALLS    :      wprintf
                 getche
                 continue_lsn
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard & Keith Calcote
 PURPOSE  :      Allows the user to choose any page in the lesson to view.

************************************************************************/
static void enter_page(void)
{
  /*
    Initialize some temporary variables.
  */
  temp = 0 ;
  temp1 = recno ;
  wprintf("\rEnter page number: ") ;
  while( (ch = getche()) != 13)
  {
     if ( ch <= 57 && ch >= 48 )
    {
      temp = temp*10 + ch-48 ;
      recno = temp ;
    }
    else
      {
      wprintf("  NOT A VALID PAGE NUMBER \n");
      recno = temp1 ;
      break ;
    }
  }

   if(recno > length[0])
    recno = 1 ;

 /*
    Display the selected page.
  */
  continue_lsn();
}
```

```
/**********************************************************************

 FUNCTION :      notebook
 CALLED BY:      lsn
 CALLS   :       spawnl
                 top_bar
                 lsn_bar
                 wactiv
                 wborder
                 continue_lsn
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard
 PURPOSE  :      Suspends the lsn.exe program and calls the dmt.exe program
                 to utilize the user interface's notebook functionality.

**********************************************************************/

static void notebook(void)
{
  /*
    Set up the command line for the dmt.exe program.
  */
  char *args[3];
  args[0] = "dmt.exe";
  args[1] = "dmt.exe";
  args[2] = "N";
  args[3] = NULL;

  /*
    Call the dmt.exe program.
  */
  spawnl(P_WAIT,args[0],args[1], args[2],NULL);

  /*
    Re-establish the lesson screen.
  */
  top_bar();
  lsn_bar();
  wactiv(w[1]);
  wborder(3);
  continue_lsn();
}
```

```
/***********************************************************************

 FUNCTION :       pageclr
 CALLED BY:       continue_lsn
 CALLS   :        NONE
 MODIFIED :       4/12/90
 PERSON  :        Rick Howard & Keith Calcote
 PURPOSE  :       Clears the page buffer

***********************************************************************/

static int pageclr(void)
{
  *page = '\x00' ;
}

/***********************************************************************

 FUNCTION :       page_down
 CALLED BY:       lsn
 CALLS   :        continue_lsn
 MODIFIED :       4/12/90
 PERSON  :        Rick Howard & Keith Calcote
 PURPOSE  :       Sets the lesson page number to the next page or, if the
                  current page is the last page of the lesson, sets the
                  lesson page number to page 1.

***********************************************************************/

static void page_down(void)
{
  recno ++ ;
  if(recno > length[0])
    recno = 1 ;
  continue_lsn();
}
```

```
/***********************************************************************

 FUNCTION :       page_up
 CALLED BY:       lsn
 CALLS    :       continue_lsn
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard & Keith Calcote
 PURPOSE  :       Sets the lesson page number to the previous page or, if the
                  current page is the first page of the lesson, sets the
                  lesson page number to the last page of the lesson.

***********************************************************************/

static void page_up(void)
{
  recno -- ;
  if(recno <= 0)
     recno = length[0] ;
  continue_lsn();
}
```

```
/**********************************************************************

  FUNCTION :      quit_menu
  CALLED BY:      lsn
  CALLS    :      wmenubeg
                  wmenuitem
                  wmenuend
                  wmenuget
                  error_exit
                  whelpcat
  MODIFIED :      4/12/90
  PERSON   :      Rick Howard
  PURPOSE  :      Defines the quit menu structure.

**********************************************************************/

static void quit_menu(void)
{
  int selection;  /* The user's menu choice   */

  /*
    The menu structure.
  */
  wmenubeg(2,55,6,77,0,YELLOW|_BLUE,YELLOW|_BLUE,add_shadow);
  wmenuitem(0,0,"Save Current Position",'S',70,M_CLOSE,
            save_position,0,H_SAVE_POSITION);
  wmenuitem(1,0,"Display Results",'D',71,M_CLOSE,do_nothing,0,0);
  wmenuitem(2,0,"Exit",'E',72,M_CLOSE,quit,0,H_EXIT);
  wmenuend(70,M_PD|M_SAVE,0,1,YELLOW|_BLUE,
                                        LCYAN|_BLUE,0,YELLOW|_LGREY);

  /*
    Process the menu
  */
  selection=wmenuget();
  if(selection==-1&&_winfo.errno>W_ESCPRESS) error_exit(1);
  whelpcat(H_LSN_HELP);
}
```

```
/***********************************************************************

 FUNCTION :        save_position
 CALLED BY:        quit_menu
 CALLS    :        initialize_linked_list
                   find_card
                   write_file
                   quit
 MODIFIED :        4/12/90
 PERSON   :        Rick Howard
 PURPOSE  :        Saves the user's current page number and lesson name to
                   disk so that the user may return to it later.

***********************************************************************/

static void save_position(void)
{
  int start_lsn = FALSE;      /*  Used in the find_card function indicating
                                  the user is ending a session and not
                                  beginning one                    */
  /*
    Bring in all the saved positions from disk and place into
    a linked list.
  */
  initialize_linked_list();

  /*
    Get the user's SSN.  Place the SSN, the lesson name and lesson
    page number into the linked list.
  */
  find_card(ARGS[1],ARGS[2],recno, start_lsn);

  /*
    Write the linked list to disk.
  */
  write_file("cardfile.dat");

  /*
    Exit the program.
  */
  quit();
}
```

```
/**********************************************************************

 FUNCTION :      tools
 CALLED BY:      lsn
 CALLS   :       spawnl
                 top_bar
                 lsn_bar
                 wactiv
                 wborder
                 continue_lsn
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE  :      Suspends the lsn.exe program and calls the dmt.exe program
                 to utilize the user interface's tools functionality.

**********************************************************************/

static void tools(void)
{
  /*
    Set up the command line for the dmt.exe program.
  */
  char *args[3];
  args[0] = "dmt.exe";
  args[1] = "dmt.exe";
  args[2] = "T";
  args[3] = NULL;

  /*
    Call the dmt.exe program.
  */
  spawnl(P_WAIT,args[0],args[1], args[2],NULL);

  /*
    Re-establish the lesson screen.
  */
  top_bar();
  lsn_bar();
  wactiv(w[1]);
  wborder(3);
  continue_lsn();

}
```

APPENDIX G

THE CODE: FILE "UTIL.H"

```
/*********************************************************************
                 The Discrete Math Tutor (DMT)
            Thesis Project at the Naval Postgraduate School
              1989-1990 by Keith Calcote and Rick Howard

  LIBRARY CALLS:
                chgonkey          CXL Lib
                clearkeys         CXL Lib
                error_exit        DMT Utilities
                execl             Tubro C Lib
                exit              Turbo C Lib
                fclose            Turbo C Lib
                fopen             Turbo C Lib
                fprintf           Turbo C Lib
                gotoxy            Turbo C Lib
                hidecur           CXL Lib
                printf            Turbo C Lib
                return            Turbo C Lib
                showcurs          CXL Lib
                spawnl            Turbo C Lib
                srestore          CXL Lib
                waitkey           CXL Lib
                wcenters          CXL Lib
                wclose            CXL Lib
                wcloseall         CXL Lib
                wgetchf           CXL Lib
                whelpcat          CXL Lib
                winpbeg           CXL Lib
                winpdef           CXL Lib
                winpread          CXL Lib
                wmenubeg          CXL Lib
                wmenuend          CXL Lib
                wmenuget          CXL Lib
                wmenuitem         CXL Lib
                wmessage          CXL Lib
                wopen             CXL Lib
```

LIBRARY CALLS (CONTINUED):

wpickstr	CXL Lib
wprintf	CXL Lib
wprints	CXL Lib
wputs	CXL Lib
wreaderrs	CXL Lib
wshadow	CXL Lib
wtitle	CXL Lib

PROGRAM CALLS:

lsn.exe

UTIL FUNCTIONS:

add_shadow
add_to_notebook
begin_lsn
confirm_quit
defnotebook
defprint
do_nothing
error_close_file
error_empty_ssn
error_exit
error_open_file
error_ssn
get_last_lsn
information
interface_bar
introduction_bar
normal_exit
open_notebook
pick_algorithm
pickdef
pre_help
pre_pick1
press_a_key
quit
logic_lsn
lsn_bar
memory
top_bar
you_selected

```
COMPLETED:     4/12/90

PERSONS:       Rick Howard

PURPOSE:       Provides utility functions that both the dmt.exe and lsn.exe
               programs use.

***********************************************************************/
```

```
/* Constants */

#define CLR "\x1B[2J"
#define NBYTES 128
#define SSNSIZE 15
#define LSN_LENGTH_SIZE 20
#define LSN_NAME_SIZE 20
#define LSN_PAGE_NUM_SIZE 5
#define TRUE 1
#define FALSE 0
/*------------------------------------------------------------------*/

/* function prototypes */

static void add_shadow(void);
static void add_to_notebook(void);
static void begin_lsn(void);
static void confirm_quit(void);
static void defnotebook(void);
static void defprint(void);
static void do_nothing(void);
static void error_close_file(char name[12]);
static void error_empty_ssn(void);
static void error_exit(int errnum);
static void error_open_file(char name[12]);
static void error_ssn(void);
static void get_last_lsn(void);
static void information(void);
static void interface_bar(void);
static void introduction_bar(void);
static void normal_exit(void);
static void open_notebook(void);
static void pick_algorithm(void);
static void pickdef(void);
static void pre_help(void);
static void pre_pick1(void);
static void press_a_key(int wrow);
static void quit(void);
static void logic_lsn(void);
static void lsn_bar(void);
static void memory(void);
static void top_bar(void);
static void you_selected(char *str);
```

```
/* Globals */

static char *error_text[]= {
   NULL, /* ermum =  0, no error     */
   NULL, /* ermum == 1, windowing error */
   "error"
      "Can not find the notebook"
};
/*--------------------------------------------------------------*/
```

```
/*****************************************************************

 FUNCTION :     add_shadow
 CALLED BY:     begin_lsn          in util.h
                confirm_quit       in util.h
                defnotebook        in util.h
                error_close_file   in util.h
                error_empty_ssn    in util.h
                error_open_file    in util.h
                error_ssn          in util.h
                exams              in dmt.exe
                get_ssn            in lsn.exe
                information        in util.h
                logic_exam         in dmt.exe
                notebook           in dmt.exe
                open_notebook      in util.h
                open_title_wind    in dmt.exe
                pickdef            in util.h
                pre_help           in util.h
                pre_menu1          in dmt.exe
                pre_pick1          in util.h
                print_notebook     in dmt.exe
                quit_menu          in dmt.exe
                quit_menu          in lsn.exe
                tools              in dmt.exe
                view_notebook      in dmt.exe
 CALLS   :      wshadow
 MODIFIED :     4/12/90
 PERSON  :      Rick Howard
 PURPOSE :      This function will add a shadow to the active window

*****************************************************************/

static void add_shadow(void)
{
  wshadow(LGREY|_BLACK);
}
```

```
/********************************************************************

 FUNCTION :        add_to_notebook
 CALLED BY:        defnotebook in util.h
 CALLS    :        fopen
                   error_open_file
                   return
                   fprintf
                   getc
                   putc
                   fclose
                   error_close_file
 MODIFIED :        4/12/90
 PERSON   :        Rick Howard
 PURPOSE  :        This function retrieves a file name and appends the file to
                   the notebook

********************************************************************/

static void add_to_notebook()
{
  int c;              /* Holds the character values that are transfered from the
                         user's selected definition to his notebook file        */

  FILE *f;            /* Pointer to the user's selected file                 */

  /*
    Get the name of the user's notebook.
  */
  open_notebook();
```

```
    /*
      Based on the user's selected defintion, append that definition
      file to the user's notebook.
    */
    switch (definitions[def_number][0]){

      case GRAPH :
      {
        if ((f = fopen("graph.def", "r")) == NULL)
        {
          error_open_file("gaph.def");
          return;
        }
        fprintf(current_notebook,"\n \n");
        while((c = getc(f)) != EOF) putc(c,current_notebook);
        if (fclose(f) == EOF)
          error_close_file("graph.def");
        break;
      }
    default :
      break;
    }
    if (fclose(current_notebook) == EOF)
      error_close_file(notebook_name);
  }
```

```
/***********************************************************************

 FUNCTION :       begin_lsn
 CALLED BY:       initialize      in dmt.exe
                  main            in lsn.exe
 CALLS    :       wmenubeg
                  wmenuitem
                  wmenuend
                  wmenuget
                  error_exit
                  whelpcat
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard
 PURPOSE  :       Displays the Begin menu

***********************************************************************/

static void begin_lsn (void)
{
  int selection;  /* The user's menu choice */

  wmenubeg(2,3,5,25,0,YELLOW|_BLUE,YELLOW|_BLUE,add_shadow);
  wmenuitem(0,0,"Start a Lesson",'S',20,0,do_nothing,0,H_START_LSN);

    wmenubeg(6,3,8,9,0,YELLOW|_BLUE,YELLOW|_BLUE,add_shadow);
    wmenuitem(0,0,"Logic",'L',31,M_CLOSE,logic_lsn,0,H_LOGIC);
    wmenuend(31,M_PD|M_SAVE,0,1,YELLOW|_BLUE,
                                        LCYAN|_BLUE,0,YELLOW|_LGREY);

  wmenuitem(1,0,"Return to Last Lesson",'R',21,M_CLOSE,
           get_last_lsn,0,H_RETURN_TO_LAST_LSN);
  wmenuend(20,M_PD|M_SAVE,0,1,YELLOW|_BLUE,
                                       LCYAN|_BLUE,0,YELLOW|_LGREY);

  selection=wmenuget();
  if(selection==-1&&_winfo.errno>W_ESCPRESS) error_exit(1);
  whelpcat(H_USER_INTERFACE);
}
```

```
/*********************************************************************

 FUNCTION :       confirm_quit
 CALLED BY:       initialize          in dmt.exe
                  quit_menu           in dmt.exe
                  pre_help            in util.h
                  press_a_key         in util.h
 CALLS   :        chgonkey
                  wopen
                  error_exit
                  add_shadow
                  wputs
                  clearkeys
                  showcurs
                  wgetchf
                  normal_exit
                  wclose
                  hidecurs
                  wprintf
 MODIFIED :       4/12/90
 PERSON  :        Rick Howard
 PURPOSE  :       This function pops open a window and confirms that the user
                  really wants to quit the demo. If so, it terminates
                  the demo program.

*********************************************************************/

static void confirm_quit(void)
{
  struct _onkey_t *kblist; /* Pointer to the list of active hot-keys   */

  /*
    Set a pointer to the hot-key listfor future reference and
    disable all the hot keys.
  */
  kblist=chgonkey(NULL);
```

```
    /*
      Open the message window.
    */
    if(!wopen(9,26,13,55,0,WHITE|_BROWN,WHITE|_BROWN)) error_exit(1);
    add_shadow();
    wputs("\n Quit DMT, are you sure? \033A\156Y\b");
    clearkeys();
    showcur();

    /*
      If the user wants to exit, terminate the program. If not,
      reactivate the hoy-key list and close the window.
    */
    if(wgetchf("YN",'Y')=='Y') normal_exit();
    wclose();
    hidecur();
    chgonkey(kblist);
    wprintf("%d\n", coreleft());
}
```

```
/**********************************************************************

 FUNCTION :       defnotebook
 CALLED BY:       pickdef              in util.h
 CALLS    :       wmenubeg
                  wmenuitem
                  wmenuend
                  wmenuget
                  error_exit
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard
 PURPOSE  :       Displays the Notebook menu

**********************************************************************/

static void defnotebook(void)
{
  int choice; /* The user's menu choice */

  wmenubeg(18,25,21,52,0,YELLOW|_BLUE,YELLOW|_BLUE,add_shadow);
  wmenuitem(0,0,"Add Definition to Notebook",'A',20,M_CLOSE,
            add_to_notebook,0,0);
  wmenuitem(1,0,"Print Definition",'P',21,M_CLOSE,defprint,0,0);
  wmenuend(20,M_VERT,0,0,YELLOW|_BLUE,
                             LCYAN|_BLUE,0,YELLOW|_LGREY);

  choice = wmenuget();
  if(choice == -1 && _winfo.ermo> W_ESCPRESS) error_exit(1);
}
```

```
/**********************************************************************

 FUNCTION :      defprint
 CALLED BY:      defnotebook          in util.h
 CALLS   :       spawnl
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE  :      Sends the user selected definition file to the program
                 print.exe for printing

**********************************************************************/

static void defprint(void)
{
  switch (definitions[def_number][0]){

  case GRAPH :
    spawnl(P_WAIT,"print.exe", "print.exe", "graph.def", NULL);
    break;

  default :
    break;
  }
}
```

```
/*************************************************************************

 FUNCTION :      do_nothing
 CALLED BY:      quit_menu           in dmt.exe
                 tools               in dmt.exe
                 quit_menu           in lsn.exe
                 begin_lsn           in util.h
                 information         in util.h
 CALLS   :       NONE
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE  :      This function is used as a dummy function for
                 several menu items in the pull-down demo

*************************************************************************/

static void do_nothing(void)
{
}
```

```
/******************************************************************

 FUNCTION :        error_close_file
 CALLED BY:        add_to_notebook      in dmt.exe
 CALLS    :        wopen
                   error_exit
                   add_shadow
                   wprintf
                   wclose
 MODIFIED :        4/12/90
 PERSON   :        Rick Howard
 PURPOSE  :        Error message if the system can not properly close a file

******************************************************************/

static void error_close_file(char name[12])
{
   if (!wopen(15,24,19,57,0,WHITE|_CYAN,WHITE|_CYAN)) error_exit(1);
   add_shadow();
   wprintf("\n       Can not close file");
   wprintf("\n            %s ", name);
   wprintf("\n \n      Press Esc to Continue");
   wgetchf("'033'", 0);
   wclose();

}
```

```
/**********************************************************************

 FUNCTION :      error_empty_ssn
 CALLED BY:      find_card           in link.c
 CALLS    :      wopen
                 error_exit
                 wtitle
                 add_shadow
                 wprints
                 wgetchf
                 wclose
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard
 PURPOSE  :      Error message when the user attempts to continue a lesson
                 and there exists no students in the linked list

**********************************************************************/

static void error_empty_ssn(void)
{
   if (!wopen(15,24,20,58,0,WHITE|_CYAN,WHITE|_CYAN)) error_exit(1);
   wtitle("[ Error Window ]",TCENTER,LGREEN|_MAGENTA);
   add_shadow();
   wprints(1,3,BLINK|YELLOW|_BROWN,"No students are in the list");
   wprints(3,7,YELLOW|_BROWN,"Press Esc to continue");
   wgetchf("'\033'",'Y');
   wclose();
}
```

```
/*******************************************************************

 FUNCTION :      error_exit
 CALLED BY:      begin_lsn           in util.h
                 confirm_quit        in util.h
                 defnotebook         in util.h
                 error_close_file    in util.h
                 error_empty_ssn     in util.h
                 error_open_file     in util.h
                 error_ssn           in util.h
                 exams               in dmt.exe
                 get_ssn             in link.c
                 information         in util.h
                 logic_exam          in dmt.exe
                 main                in calc.exe
                 main                in lsn.exe
                 main                in table.exe
                 main_menu           in dmt.exe
                 menudemo            in dmt.exe
                 notebook            in dmt.exe
                 open_back_wind      in dmt.exe
                 open_notebook       in util.h
                 open_titl_wind      in dmt.exe
                 open_window         in exam.exe
                 pickdef             in util.h
                 print_notebook      in dmt.exe
                 quit_menu           in dmt.exe
                 quit_menu           in lsn.exe
                 tools               in dmt.exe
                 view_notebook       in dmt.exe
 CALLS    :      wprintf
                 wernmsg
                 exit
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard
 PURPOSE  :      Displays an appropriate error message for known problems

*******************************************************************/
```

```
static void error_exit(int errnum)
{
   if(errnum) {
      wprintf("\n%s\n",(errnum==1)?werrmsg():error_text[errnum]);
      exit(errnum);
   }
}
```

```
/**********************************************************************

 FUNCTION :        error_open_file
 CALLED BY:        print_notebook        in dmt.exe
                   view_notebook         in dmt.exe
                   add_to_notebook       in util.h
 CALLS   :         wopen
                   error_exit
                   add_shadow
                   wprintf
                   wgetchf
                   wclose
 MODIFIED :        4/12/90
 PERSON  :         Rick Howard
 PURPOSE  :        Error messagee if the system can not properly open a file

**********************************************************************/

static void error_open_file(name)
char name[12];
{
   if (!wopen(15,24,19,57,0,WHITE|_CYAN,WHITE|_CYAN)) error_exit(1);
   add_shadow();
   wprintf("      Can not open file");
   wprintf("\n             %s ", name);
   wprintf("\n       Press Esc to continue");
   wgetchf("\033'",'Y');
   wclose();

}
```

```
/**********************************************************************

 FUNCTION :      error_ssn
 CALLED BY:      get_last_lsn          in util.h
 CALLS   :       wopen
                 error_exit
                 add_shadow
                 wprints
                 wtitle
                 wgetchf
                 wclose
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard
 PURPOSE  :      Error message if the user chosen ssn is not in the student
                 linked list

**********************************************************************/

static void error_ssn(void)
{
   if (!wopen(15,24,20,58,0,WHITE|_CYAN,WHITE|_CYAN)) error_exit(1);
   wtitle("[ Error Window ]",TCENTER,LGREEN|_MAGENTA);
   add_shadow();
   wprints(1,7,BLINK|YELLOW|_BROWN,"Can not find that SSN");
   wprints(3,7,YELLOW|_BROWN,"Press Esc to continue");
   wgetchf("\033'",'Y');
   wclose();

}
```

```
/**********************************************************************

 FUNCTION :      get_last_lsn
 CALLED BY:      begin_lsn              in util.h
 CALLS   :       whelpcat
                 initialize_linked_list
                 find_card
                 error_ssn
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE :       Reads the student linked list in from a file and looks
                 for a specific ssn

**********************************************************************/

static void get_last_lsn(void)
{
  char *ARGS[3];              /* Placeholder for the function: find_card */

  int page;                   /* Indicates # of students in the linked list */

  int start_lsn = TRUE;       /* This function is always executed at the
                                 when the user begins a lesson          */

  whelpcat(H_SSN);
  initialize_linked_list();
  page = find_card(ARGS[1], ARGS[2], page, start_lsn);
  if (page == 0){
    error_ssn();
    return;
  }
}
```

```
/*********************************************************************

 FUNCTION :     information
 CALLED BY:     initialize          in dmt.exe
                initialize          in lsn.exe
 CALLS    :     whelpcat
                wmenubeg
                wmenuitem
                wmenuend
                wmenuget
                error_exit
 MODIFIED :     4/12/90
 PERSON   :     Rick Howard
 PURPOSE  :     Defines the information menu structure

*********************************************************************/

static void information(void)
{
  int selection; /* The user's menu choice */

  wmenubeg(2,13,6,26,0,YELLOW|_BLUE,YELLOW|_BLUE,add_shadow);
  wmenuitem(0,0,"Definitions",'D',40,M_CLOSE,pickdef ,0,H_DEFINITIONS);
  wmenuitem(1,0,"Examples",'E',41,0,do_nothing,0,H_EXAMPLES);
  wmenuitem(2,0,"Theorems",'T',42,0,do_nothing,0,H_THEOREMS);
  wmenuend(40,M_PD|M_SAVE,0,1,YELLOW|_BLUE,
                                   LCYAN|_BLUE,0,YELLOW|_LGREY);

  selection=wmenuget();
  if(selection==-1&&_winfo.errno>W_ESCPRESS) error_exit(1);
  whelpcat(H_USER_INTERFACE);
}
```

```
/**********************************************************************

 FUNCTION :       interface_bar
 CALLED BY:       menudemo              in dmt.exe
 CALLS    :       wopen
                  wprints
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard
 PURPOSE  :       Displays the bottom screen help bar for the dmt interface

**********************************************************************/

static void interface_bar(void)
{
   char help[]="H-Help";
   char exit[]="ESC-Back up";

   wopen(24,0,25,79,5,YELLOW|_BLUE,YELLOW|_BLUE);
   wprints(0,1,LCYAN|_BLUE,help);
   wprints(0,68,LCYAN|_BLUE,exit);
}
```

```
/**********************************************************************

 FUNCTION :       introduction_bar
 CALLED BY:       main in dmt.exe
 CALLS    :       wopen
                  wprints
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard
 PURPOSE  :       Displays the bottom screen help bar for the introduction
                  screen to the dmt interface

**********************************************************************/

static void introduction_bar(void)
{
   char help[]="H-Help";
   char exit[]="ESC-Quit";

   wopen(24,0,25,79,5,YELLOW|_BLUE,YELLOW|_BLUE);
   wprints(0,1,LCYAN|_BLUE,help);
   wprints(0,72,LCYAN|_BLUE,exit);
}
```

```
/**********************************************************************

  FUNCTION :     normal_exit
  CALLED BY:     main              in dmt.exe
                 confirm_quit      in dmt.exe
  CALLS    :     srestore
                 gotoxy
                 showcur
                 exit
  MODIFIED :     4/12/90
  PERSON   :     Rick Howard
  PURPOSE  :     This function handles normal termination.  The original
                 screen and cursor coordinates are restored before exiting
                 to DOS with ERRORLEVEL 0.

**********************************************************************/

static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  showcur();
  exit(0);
}
```

```
/*********************************************************************

 FUNCTION :      open_notebook
 CALLED BY:      add_to_notebook      in dmt.exe
 CALLS    :      chgonkey
                 wopen
                 error_exit
                 add_shadow
                 wtitle
                 winpbeg
                 wprints
                 winpdef
                 winpread
                 wputs
                 wgetchf
                 wclose
                 findfirst
                 strcmp
                 fopen
                 hidecurs
                 error_exit
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard
 PURPOSE  :      Asks the user for his notebook name and sets the variable
                 current_notebook = to it.

*********************************************************************/

static void open_notebook()
{
  struct ffblk ffblk;          /* Place holder for the function findfirst    */

  struct _onkey_t *k1;         /* Pointer to current hot-key list     */

  int done;                    /* Indicates if the function findfirst
                                  found the user's notebook name in the
                                  current directory               */

  static char file_operation;  /* Indicates if the user wants to append
                                  to an existing notebook file or
                                  create another one              */

  register int response;       /* The user's response             */
```

```
/*
   Assign the current hot key list to k1 and set the current hot
   key list to NULL.
*/
k1 = chgonkey(NULL);

/*
   Open a window to retrieve the user's notebook name.
*/
if(!wopen(10,8,17,70,1,LCYAN|_BLUE, LCYAN|_BLUE)) error_exit(1);
add_shadow();
wtitle("[Name Your Personalized Notebook]",TLEFT, LCYAN|_BLUE);

/* Display prompts and define fields. */
do{
   winpbeg(LGREEN|_LGREY,WHITE|_LGREY);

   wprints( 1, 3, WHITE|_BLUE, "What is your Notebook Name?");
   winpdef( 1, 35, notebook_name, "WWWWWWWWWWWW",0,0,NULL,0);
   wprints( 3, 3, WHITE|_BLUE, "(A)ppend or (O)verwrite:");
   winpdef( 3, 35, file_operation, "<AaOo>",0,0,NULL,0) ;

   /*
      Mark end of form and process it.
   */
   if(winpread()) break;

   if (!wopen(15,24,19,57,0,WHITE|_CYAN,WHITE|_CYAN)) error_exit(1);
   add_shadow();
   wputs("\n Is this information correct? \033A\076Y\b");
   response = wgetchf("YN",'Y');

   wclose();
}
while (response != 'Y');

/*
   Re-enable the hot-key list.
*/
chgonkey(k1);

/*
```

```
    Look for the user's notebook file in the current directory.
  */
  done = findfirst(notebook_name, &ffblk, 0);

  /*
    If the file is found and the user wishes to append to it, open the
    file appropriately.  If the file is not found or the user wishes to
    overwrite it, open it appropriately.
  */
  if (done == 0)
    if ((strcmp(file_operation,"A") == 0) ||
      (strcmp(file_operation,"a") == 0))
          current_notebook = fopen(notebook_name, "a");
  else
    current_notebook = fopen(notebook_name, "w+t");
  else
    current_notebook = fopen(notebook_name, "w+t");

  wclose();
  hidecur();
}
```

```
/*********************************************************************

 FUNCTION :       pickdef
 CALLED BY:       information          in dmt.exe
 CALLS   :        wopen
                  error_exit
                  add_shadow
                  whelpcat
                  wprintf
                  wpickstr
                  pre_pick1
                  you_selected
                  defnotebook
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard
 PURPOSE  :       Allows the user to choose a definition from a list of
                  definitions and add it to his personalized notebook

*********************************************************************/

static void pickdef(void)
{
   int def_number;     /* Indicates the array placement of the user
                          chosen definition                        */

   /*
      OPen a window for the definition choices.
   */
   if(!wopen(10,11,17,68,3,LMAGENTA|_RED,LRED|_MAGENTA)) error_exit(1);
   add_shadow();
   whelpcat(H_SELECT);
   wprintf("\033R\001\033C\003Select a definition =>\033R\001\033C\003");

   /*
      Allow the user to choose a definition.
   */
   def_number = wpickstr(6,32,11,-1,0,
                         LGREEN|_RED,LCYAN|_RED,RED|_LGREY,
                         definitions,0,pre_pick1);
```

```
    /*
       Show the user the definition he chose.
    */
    you_selected(definitions[def_number]);

    /*
       Give the user the option to add the definition to the notebook or
       print it out.
    */
    defnotebook( ):

    wclose();
}
```

```
/*********************************************************************

 FUNCTION :     pre_help
 CALLED BY:     initialize          in dmt.exe
                main                in lsn.exe
 CALLS    :     add_shadow
                setonkey
 MODIFIED :     4/12/90
 PERSON   :     Rick Howard
 PURPOSE  :     Adds a shadow to all help screens

*********************************************************************/

static void pre_help(void)
{
  add_shadow();
}

/*********************************************************************

 FUNCTION :     pre_pick1
 CALLED BY:     pickdef in util.h
 CALLS    :     wmessage
                add_shadow
 MODIFIED :     4/12/90
 PERSON   :     Rick Howard
 PURPOSE  :     Sets up the window for shadow and borders

*********************************************************************/

static void pre_pick1(void)
{
  wmessage("? ?",BT_BORD,4,LGREEN|_RED);
  add_shadow();
}
```

```
/*********************************************************************

 FUNCTION :       press_a_key
 CALLED BY:       you_selected          in util.h
 CALLS    :       wcenters
                  hidecurs
                  waitkey
                  confirm_quit
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard
 PURPOSE  :       Displays a message to press any key to continue and
                  waits for the user to follow the instruction.

*********************************************************************/

static void press_a_key(int wrow)
{
  register int attr; /* The scrren attribute for the msg window      */

  attr=(BLINK|YELLOW)|((_winfo.active->wattr>>4)<<4);
  wcenters(wrow,attr,"Press a key");
  hidecur();
  if(waitkey()==ESC) confirm_quit();
  wcenters(wrow,attr,"           ");
}
```

```
/************************************************************************

 FUNCTION :      quit
 CALLED BY:      quit_menu          in lsn.exe
                 save_position      in link.c
 CALLS   :       wcloseall
                 printf
                 exit
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE  :      Terminates the program by closing all windows, clearing
                 the screeen and exiting the program

************************************************************************/

static void quit(void)
{
  wcloseall();
  printf(CLR);
  exit(1);
}
```

```
/*********************************************************************

 FUNCTION :       logic_lsn
 CALLED BY:       begin_lsn            in util.h
 CALLS    :       execl
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard
 PURPOSE  :       Terminate the dmt.exe program and execute the lsn.exe
                  program

*********************************************************************/

static void logic_lsn(void)
{
  char *args[]={
    "lsn.exe","logic.len","logic.txt"    };  /* The command line for
                                               lsn.exe program    */

  execl("lsn.exe",args[0],args[1],args[2],NULL);
}
```

```
/***********************************************************************

 FUNCTION :      lsn_bar
 CALLED BY:      main              in lsn.exe
                 notebook          in lsn.exe
                 tools             in lsn.exe
 CALLS    :      wopen
                 wprints
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard
 PURPOSE  :      Displays the bottom screen help window for any lsn

***********************************************************************/

static void lsn_bar(void)
{
   char help[]="H-Help";
   char find[]="ALT P-Find Page #";
   char exit[]="ESC-Back up";

   wopen(24,0,25,79,5,YELLOW|_BLUE,YELLOW|_BLUE);
   wprints(0,1,LCYAN|_BLUE,help);
   wprints(0,31.YELLOW|_BLUE.find);
   wprints(0,68.LCYAN|_BLUE,exit);
}
```

```
/*********************************************************************

 FUNCTION :      top_bar
 CALLED BY:      menudemo         in dmt.exe
                 main             in lsn.exe
                 notebook         in lsn.exe
                 tools            in lsn.exe
 CALLS   :       wopen
                 wprints
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE  :      Displays the top screen help window for the interface

*********************************************************************/

static void top_bar(void)
{
   char begin1[]="B",begin2[]="egin";
   char information1[]="I",information2[]="nformation";
   char exams1[]="E",exams2[]="xams";
   char tools1[]="T",tools2[]="ools";
   char notebook1[]="N",notebook2[]="otebook";
   char quit1[]="Q",quit2[]="uit";

   wopen(0,0,2,79,0,YELLOW|_BLUE,YELLOW|_BLUE);
   wprints(0,2,LCYAN|_BLUE,begin1);
   wprints(0,3,YELLOW|_BLUE,begin2);
   wprints(0,12,LCYAN|_BLUE,information1);
   wprints(0,13,YELLOW|_BLUE,information2);
   wprints(0,30,LCYAN|_BLUE,exams1);
   wprints(0,31,YELLOW|_BLUE,exams2);
   wprints(0,42,LCYAN|_BLUE,tools1);
   wprints(0,43,YELLOW|_BLUE,tools2);
   wprints(0,56,LCYAN|_BLUE,notebook1);
   wprints(0,57,YELLOW|_BLUE,notebook2);
   wprints(0,72,LCYAN|_BLUE,quit1);
   wprints(0,73,YELLOW|_BLUE,quit2);
}
```

```
/**********************************************************************

 FUNCTION :      you_selected
 CALLED BY:      pickdef              in util.h
 CALLS    :      wprintf
                 wreadcur
                 press_a_key
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard
 PURPOSE  :      This function is used by the List Picking demo to display
                 a selected string, or display an error message if an error
                 occurred.  It also prompts the user for a keypress 2 lines
                 below the string/error message.

**********************************************************************/

static void you_selected(char *str)
{
  int wrow,wcol;

  if(_winfo.errno)
    wprintf("\033EL%s",werrmsg());
  else
    wprintf("\033ELYou selected:  \033F\005%s\033F\004",str);
  wreadcur(&wrow,&wcol);
  press_a_key(wrow+2);
}
```

APPENDIX H

THE CODE: FILE "CALC.C"

```
/**********************************************************************
                      The Discrete Math Tutor (DMT)
                Thesis Project at the Naval Postgraduate School
                  1989-1990 by Keith Calcote and Rick Howard

 LIBRARY CALLS:
               error_exit        DMT Utilities
               setonkey          CXL Lib
               set_video         DMT Utilities
               wcclear           CXL Lib
               wcenters          CXL Lib
               wclear            CXL lib
               wgotoxy           CXL Lib
               whelpdef          CXL Lib
               whline            CXL lib
               wopen             CXL Lib
               wprintf           CXL Lib
               wscanf            CXL Lib
               wshadow           DMT Utilities
               wslide            CXL Lib
               wtitle            CXL Lib

 PROGRAM CALLS:
               NONE

 CALC FUNCTIONS:
               display_loop
               divv
               error_msg
               minus
               mult
               pre-help
               plus
               quit
               refresh
               table
```

```
COMPLETED: 4/12/90

PERSONS: Rick Howard

PURPOSE: Provide a simple four function calculator

************************************************************************/
```

```
/* header files */

#include <stdlib.h>
#include <math.h>
#include "d:\cxl\cxlwin.h"
#include "d:\cxl\cxlkey.h"
#include "d:\cxl\cxlvid.h"
#include "d:\cxl\cxlstr.h"
#include "d:\tc\thesis\video.h"
#include "d:\tc\thesis\help.h"
/*----------------------------------------------------------------------*/

/* function prototypes */

static int chk_data_fld (char *input_field);
static void confirm_quit(void);
static unsigned get_key(int *done);
static void quit(void);
static void plus(void);
static void minus(void);
static void mult(void);
static void divv(void);
static void display_loop(void);
static void error_msg(int type);
static void reset(void);
static void refresh(void);
static void table(void);
static void pre_help(void);
static void input(void);

/*----------------------------------------------------------------------*/
```

```
/* constants */

#define INVALID_NUMBER 1
#define INVALID_OPERATOR 2
#define RESET1 .999
#define RESET2 .888
#define RESET3 "Q"
#define MAXNUM 10e36
#define MINNUM 10e-36
#define TRUE 1
#define FALSE 0
#define ZERODIV 0
#define BADOP 1
#define BAD_DATA_FLD 3

/*------------------------------------------------------------------*/

/* globals */

double num1, num2, answer;          /*  The two operands and the result */

char op[1];                         /* The operator */

char num1str[35], num2str[35];      /*  The input operands */

WINDOW w;                           /* The window handle */

int i = 0;                          /* Counter */

int invalid_expression = 0;         /* Used to determine invalid input */

unsigned key;                       /* Define alternate keyboard get function. */

int division_error = FALSE;         /* Used to determine division by zero error */
```

```
/**********************************************************************

 FUNCTION :      main
 CALLED BY:      NONE
 CALLS   :       See Declarations
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE  :      See Declarations

**********************************************************************/

main()
{
  /*
    Check for mono, CGA or EGA screen.
  */
  set_video();

  /*
    Set up the hot-key to quit the program.
  */
  setonkey(0x011B,confirm_quit,0); /* ESC */

  /*
    Open a window for the calculator.
  */
  if((w=wopen(2,0,20,55,3,WHITE|_CYAN,WHITE|_CYAN))==0)
    wprintf("error_exit(1);");
  wtitle("[DMT Calculator]",TCENTER, WHITE|_CYAN);
  wmessage("[ESC-QUIT]",BT_BORD,4,YELLOW|_CYAN);
  wmessage("[H-HELP]",BT_BORD,45,YELLOW|_CYAN);
  wshadow(LGREY|_BLACK);
  wslide(2,25);
  wslide(2,10);

  /*
    Define the help screen attributes.
  */
  whelpdef("DMT.HLP",0x2368,BLACK|_LGREY,BLACK|_LGREY,
                                 LBLUE|_LGREY,LRED|_LGREY,pre_help);
  whelpcat(H_CALCULATOR_HELP);
```

```
while (1){

   /*
      Initial set up.
   */
   invalid_expression = FALSE;
   wcclear(WHITE|_CYAN);

   /*
      Accept input from the user.
   */
   input();

   /*
      Perform the appropriate calculation based upon the operator.
   */
   switch (*op){
   case '+':
      plus();
      break;
   case '-':
      minus():
      break;
   case '*':
      mult();
      break;
   case '/':
      divv();
      break;
   default:
      error_msg(BADOP);
   }
```

```
        /*
            Check for division error and continue.
        */
        if (division_error == FALSE){
            whelpcat(H_CALC7);
            wcenters(15,YELLOW|_LGREY,"Press any key to continue");
            waitkey();
        }
        refresh();
    }
}
```

```
/**********************************************************************

 FUNCTION :     quit
 CALLED BY:     calc
 CALLS    :     exit
                wcloseall
                clrscr
 MODIFIED :     4/12/90
 PERSON   :     Rick Howard
 PURPOSE  :     Terminates the program

**********************************************************************/

static void quit(void)
{
  wcloseall();
  clrscr();
  exit(0);
}

/**********************************************************************

 FUNCTION :     plus
 CALLED BY:     calc
 CALLS    :     wprintf
                table
                gotoxy
 MODIFIED :     4/12/90
 PERSON   :     Rick Howard
 PURPOSE  :     Performs the addition operation on both operands and displays
                the results.

**********************************************************************/

static void plus(void)
{
  wgotoxy(6,5);
  answer = num1 + num2;
  wprintf("%30.2f", answer);
  table();
}
```

```
/**********************************************************************

 FUNCTION :       minus
 CALLED BY:       calc
 CALLS    :       wprintf
                  table
                  gotoxy
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard
 PURPOSE  :       Performs the subtraction operation on both operands
                  and displays the results.

**********************************************************************/

static void minus(void)
{
   wgotoxy(6,5);
   answer = num1 - num2;
   wprintf("%30.2f", answer);
   table();
}

/**********************************************************************

 FUNCTION :       mult
 CALLED BY:       calc
 CALLS    :       wprintf
                  table
                  gotoxy
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard
 PURPOSE  :       Performs the multiplication operation on both operands
                  and displays the results.

**********************************************************************/

static void mult(void)
{
   wgotoxy(6,5);
   answer = num1 * num2;
   wprintf("%30.2f", answer);
   table();
}
```

```
/************************************************************************

 FUNCTION :        divv
 CALLED BY:        calc
 CALLS    :        wprintf
                   table
                   gotoxy
 MODIFIED :        4/12/90
 PERSON   :        Rick Howard
 PURPOSE  :        Performs the division operation on both operands
                   and displays the results.

************************************************************************/

static void divv(void)
{
  wgotoxy(6,5);
  if (num2 == 0.0)
    error_msg(ZERODIV);
  else
    {
    reset();
    answer = num1 /num2;
    wprintf("%30.2f", answer);
    table();
  }
}
```

```
/***********************************************************************

 FUNCTION :      error_msg
 CALLED BY:      calc
 CALLS   :       wcenters
                 wopen
                 wprintf
                 wshadow
                 whelpcat
                 waitkey
                 wclose
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE :       Displays an error message for any recognized invalid
                 expression.

***********************************************************************/

static void error_msg(int type)
{
   int ch; /* Holds user input */

   /*
      Open error window.
   */
   if(!wopen(13,20,17,53,0,WHITE|_RED,WHITE|_RED)) wprintf("error_exit(1)");
   wshadow(DGREY|_BLACK);

   switch (type){
   case 0:
      whelpcat(H_CALC6);
      wcenters(0,BLINK|WHITE|_RED,"Division By Zero");
      break;
   case 1:
      whelpcat(H_CALC8);
      wcenters(0,BLINK|WHITE|_RED,"Invalid Operator");
      break;
   case 3:
      whelpcat(H_CALC9);
      wcenters(0,BLINK|WHITE|_RED,"Invalid Data Field Entry");
      break;
```

```
    default:
        break;
    }

    /*
        Wait for the user's response.
    */
    wcenters(2,WHITE|_RED,"Press any key to continue");
    clearkeys();
    waitkey();
    division_error = TRUE;
    wclose();

}
```

```
/**********************************************************************

 FUNCTION :       table
 CALLED BY:       plus
                  minus
                  divv
                  mult
 CALLS   :        wgotoxy
                  wprintf
                  whline
 MODIFIED :       4/12/90
 PERSON  :        Rick Howard
 PURPOSE :        Displays the two operands, the operator and the solution
                  in a nice tabular format.

**********************************************************************/

static void table(void)
{
   division_error = FALSE;
   wgotoxy(3,5);
   wprintf("%30.2f",num1);
   wgotoxy(4,5);
   wprintf("%30.2f",num2);
   wgotoxy(4,40);
   wprintf("%c".*op);
   whline(5,12,25,0,BLACK|_CYAN);
}
```

```
/*********************************************************************

 FUNCTION :      reset
 CALLED BY:      calc
 CALLS   :       NONE
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE :       Resets the error indicator booleans

*********************************************************************/

static void reset(void)
{
  division_error = FALSE;
  invalid_expression = FALSE;
}

/*********************************************************************

 FUNCTION :      refresh
 CALLED BY:      calc
 CALLS   :       wclear
                 wgotoxy
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE :       Clears the calculator window.

*********************************************************************/

static void refresh()
{
  wclear();
  wgotoxy(15,5);
}
```

```
/*****************************************************************************

 FUNCTION :     pre_help
 CALLED BY:     calc
 CALLS    :     wshadow
 MODIFIED :     4/12/90
 ?ERSON   :     Rick Howard
 PURKPOSE :     Adds a shadow to the help screen and sets a hot-key that
                allows the user to quit the program.

*****************************************************************************/

static void pre_help(void)
{
  wshadow(LGREY|_BLACK);
  setonkey(0x2d00,quit,0);
}
```

```
/*********************************************************************

 FUNCTION :      input
 CALLED BY:      calc
 CALLS   :       wshadow
                 wopen
                 wprintf
                 wtitle
                 wmessage
                 winpbeg
                 wprints
                 winpdef
                 winpkey
                 winpread
                 wputs
                 wgetchf
                 cvtcf
                 wclose
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE  :      Allows the user to input numbers for calculations.

*********************************************************************/

static void input(void)
{
  register int ch; /* Holds user input */
  register int mode=0; /* Toggles data field category */

  /*
     Open a window for the input.
  */
  if(!wopen(4,17,18,57,1,LCYAN|_BLUE,LCYAN|_BLUE)) wprintf("error_exit(1)");
  wshadow(DGREY|_BLACK);
  wtitle("[ Calculator Input Pad ]",TCENTER,LCYAN|_BLUE);
  wmessage(" [F10]=Calculate ",BT_BORD, 12,LCYAN|_BLUE);
  do {

    /*
       Mark beginning of form.
    */
    winpbeg(LGREEN|_LGREY,WHITE|_LGREY);
    /*
       Display prompts and define fields.
    */
    wprints(2,5,WHITE|_BLUE,"First Number");
    winpdef(2,20,num1str,"999999999.99",'9',mode,chk_data_fld,H_CALC1);

    wprints(4,5,WHITE|_BLUE,"Operator");
    winpdef(4,20,op,"<*+/->",0,mode,NULL,H_CALC2);

    wprints(6,5,WHITE|_BLUE,"Second Number");
    winpdef(6.20,num2str,"999999999.99",'9',mode,chk_data_fld,H_CALC3);

    /*
        Define alternate keyboard get function.
    */
    winpkey(get_key,&key);

    /*
       Mark end of form and process it.  If [Esc] was pressed,
       then don't bother with the confirmation message.
    */
    if(winpread()) break;

    /*
       Display confirmation message.
    */
    if(!wopen(13,20,17,53,0,WHITE|_CYAN,
                                  WHITE|_CYAN)) wprintf("error_exit(1)");
    wshadow(DGREY|_BLACK);
    wputs("\n Is this infomation correct? \033A\076Y\b");
    clearkeys();
    whelpcat(H_CALC4);
    ch=wgetchf("YN",'Y');
    wclose();

    /*
       Change field mode to "update".
    */
    mode=1;

 }
 while(ch!='Y');
```

```
    /*
       Convert the input operand strings to floats.
    */
    num1 = cvtcf(num1str,9,2);
    num2 = cvtcf(num2str,9,2);

    /*
       Close the input window.
    */
    wclose();
}
```

```
/*******************************************************************

 FUNCTION :      confirm_quit
 CALLED BY:      calc
 CALLS    .      chgonkey
                 wopen
                 wprintf
                 wshadow
                 wputs
                 clearkeys
                 whelpcat
                 wgetchf
                 wclose
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard
 PURPOSE  :      Allows the user the option to quit the program or
                 continue where he left off.

*******************************************************************/

static void confirm_quit(void)
{
  struct _onkey_t *kblist;      /* Pointer to the list of active hot-keys */

  /*
    Save the hoy-key list in kblist and set the active list to NULL.
  */
  kblist=chgonkey(NULL);

  /*
    Open a message window.
  */
  if(!wopen(14,22,18,51,0,WHITE|_BROWN,
                           WHITE|_BROWN)) wprintf("error_exit(1)");
  wshadow(LGREY|_BLACK);
  wputs("\n  Quit, are you sure? \033A\156Y\b");
```

```
    /*
      Accept the user's choice.
    */
    clearkeys();
    whelpcat(H_CALC5);
    if(wgetchf("YN",'Y')=='Y') quit();
    wclose();

    /*
      Reset the active hot-key list.
    */
    chgonkey(kblist);
}
```

```
/**********************************************************************

  FUNCTION :     chk_data_fld
  CALLED BY:     calc
  CALLS    :     error_msg
  MODIFIED :     4/12/90
  PERSON   :     Rick Howard
  PURPOSE  :     Checks data strings for valid input

**********************************************************************/

static int chk_data_fld (char *input_field)
{
   int num_plus=0,num_minus=0;              /* Holds the number of operators in the
                                                input string                    */

   int current_position=1,error_position=0; /* Keeps track of position of
                                               the error in the input string */

   int error_flag = FALSE;                  /* Indicates if an error was found */

   /*
      Search for end of text.
   */
   while(*input_field++!=' ') current_position++;

   /*
      Search for the begining of the string.
   */
   while(*input_field == ' ')
      input_field++;
```

```
    /*
      Checks for multiple '+' or '-' signs.
    */
    while (*input_field != '\0'){
        if (*input_field == '+'){
            num_plus++;
            if (current_position > 1)
                error_flag = TRUE;
        }
        if (*input_field == '-'){
            num_minus++;
            if (current_position > 1)
                error_flag = TRUE;
        }
        input_field++;
        current_position++;
    }

    /*
        if more than one plus or minus return error
    */
    if((num_plus > 1) || (num_minus > 1) || (error_flag)){
        error_msg(BAD_DATA_FLD);
        return(1);
    }
    else
        return(0);
}
```

APPENDIX I

THE CODE: FILE "LINK.C"

```
/***********************************************************************
                    The Discrete Math Tutor (DMT)
               Thesis Project at the Naval Postgraduate School
                 1989-1990 by Keith Calcote and Rick Howard

 LIBRARY CALLS:
             execl            Turbo C Lib
             fclose           Turbo C Lib
             ferror           Turbo C Lib
             fgets            Turbo C Lib
             fopen            Turbo C Lib
             fprintf          Turbo C Lib
             free             Turbo C Lib
             itoa             Turbo C Lib
             malloc           Turbo C Lib
             printf           Turbo C Lib
             puts             Turbo C Lib
             strcmp           TURBO C Lib
             strcpy           Turbo C Lib
             strdup           Turbo C Lib
             strlen           Turbo C Lib
            wclose            CXL Lib
             winputsf         CXL Lib
             wopen            CXL Lib
             wtitle           CXL LIb

 PROGRAM CALLS:
             lsn.exe
```

```
LINK FUNCTIONS:
                    add_card
                    add_shadow
                    error
                    error_exit
                    find_card
                    get_ssn
                    initialize_linked_list
                    insert_node
                    list_cards
                    prt_record
                    read_file
                    write_file

COMPLETED:     4/12/90

PERSONS:       Rick Howard ( Liberally borrowed code from Augie
               Hansen's book, "C Programming: A Complete Guide
               to Mastering the C Language".)

PURPOSE:       Provides all the functionality to maintain a linked list.

*********************************************************************/
```

```
/* header files */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
/*--------------------------------------------------------------*/

/* Type Definitions */

typedef struct card_st{
   char ssn[SSNSIZE];
   char lsn_length[LSN_LENGTH_SIZE];
   char lsn_name[LSN_NAME_SIZE];
   char lsn_page_num[LSN_PAGE_NUM_SIZE];
   struct card_st *next;
}
CARD;

CARD listhead, *head, *current;
/*--------------------------------------------------------------*/

/* function prototypes */

static void get_ssn(char buf[10]);
static int initialize_linked_list();
static int add_card(char ssan[SSNSIZE], char buf1[LSN_LENGTH_SIZE],
                char buf2[LSN_NAME_SIZE], int page);
static int find_card(char buf1[LSN_LENGTH_SIZE],
                char buf2[LSN_NAME_SIZE], int page, int start_lsn);
static int read_file(char *);
static int write_file(char *);
static CARD *insert_node(CARD *);
static void error(char *);
static void prt_record(char *, char *, char *, char *);
static int list_cards(void);
/*--------------------------------------------------------------*/
```

```
/**********************************************************************

 FUNCTION :      initialize_linked_list
 CALLED BY:      save_position from the dmt.exe program
                 get_last_lsn  from the lsn.exe program
 CALLS   :       read_file
                 error
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE :       Read the student information file into a linked list
                 for manipulation.

**********************************************************************/

static int initialize_linked_list()
{
  int rc;                    /* Indicates an error after
                                execution of the read_file
                                function              */

                             /* The name of the student
                                information file      */
  static char data_file[NBYTES + 1] = {
    "cardfile.dat"
  };

  /*
    Set up the linked list pointers.
  */
  current = head = &listhead;
  head->next = head;

  /*
    Read the student information file into the linked list.
  */
  rc = read_file(data_file);
  if (rc)
    error("Cannot read data file");

  return EXIT_SUCCESS;
}
```

```
/*******************************************************************

  FUNCTION :      add_card
  CALLED BY:      find_card
  CALLS   :       insert_node
                  fprintf
                  strcpy
                  itoa
  MODIFIED :      4/12/90
  PERSON  :       Rick Howard
  PURPOSE  :      Adds a student to the linked list

*******************************************************************/

static int add_card(char ssan[SSNSIZE], char buf1[LSN_LENGTH_SIZE],
           char buf2[LSN_NAME_SIZE], int page)
{
  CARD *tmp;          /* Pointer to the newly created node in the
                         linked list                         */

  /*
    Create a new node in the linked list.
  */
  tmp = insert_node(current);
  if (tmp == NULL){
    fprintf(stderr, "Out of memory");
  }

  /*
    Place the student information into the new node.
  */
  else {
    current = tmp;
    strcpy(current->ssn, ssan);
    strcpy(current->lsn_length,buf1);
    strcpy(current->lsn_name,buf2);
    itoa(page, current->lsn_page_num,10);
  }
  return 0;
}
```

```
/*********************************************************************

 FUNCTION :      find_card
 CALLED BY:      save_position from the program lsn.exe
                 get_last_lsn  from the program dmt.exe
 CALLS   :       error_empty_ssn
                 get_ssn
                 strlen
                 trncmp
                 strcpy
                 itoa
                 free
                 add_card
                 execl
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE :       Locates a student in the student information file

*********************************************************************/

static int find_card(char buf1[LSN_LENGTH_SIZE],
            char buf2[LSN_NAME_SIZE], int page, int start_lsn)
{
  int rc = 0;          /* Indicates an error inside the list             */

  int hits = 0;        /* Indicates if the student was found in the list    */

  int len;             /* Number of characters in a SSN                  */

  char *cp;            /* Points to the SSN field in the tmp structure      */

  CARD *tmp;           /* Holds the information that needs to be found in
                          the linked list                          */

  char ssan[10];       /* Holds the SSN of the of the student to be found
                          in the linked list                       */
```

```
                /* Place holders for the command line to call the
                   lsn.exe program                          */
  char *cmds[] = {
  NULL,
  NULL,
  NULL,
  NULL,
  NULL
};

/*
  Set the tmp structure equal to the front of the list.
*/
tmp = head;
if ((tmp->next == head) && (start_lsn == TRUE)) {
  error_empty_ssn();
  return ++rc;
}

/*
  Retrieve the user's SSN.
*/
get_ssn(&ssan);
len = strlen(ssan);

/*
  For each item in the linked list...
*/
while (tmp->next != head){
  tmp = tmp->next;
  cp = tmp->ssn;
  while (*cp != '\0'){

    /*
      If the user's SSN matches the record's SSN....
    */
    if (strncmp(cp,ssan,len) == 0){
         /*
            If this is not the begining of a lesson, copy
            the user's information into the tmp structure.
         */
         if (start_lsn == FALSE){
            strcpy(tmp->lsn_length,buf1);
            strcpy(tmp->lsn_name,buf2);
            itoa(page,tmp->lsn_page_num,10);
         }

         /*
            If this is the begining of a lesson, copy the
            user's information into the command line structure.
         */
         else {

            if (cmds[2] != NULL)
               free(cmds[2]);
            cmds[2] = strdup(tmp->lsn_length);
            if (cmds[3] != NULL)
               free(cmds[3]);
            cmds[3] = strdup(tmp->lsn_name);
            if (cmds[4] != NULL)
               free(cmds[4]);
            cmds[4] = strdup(tmp->lsn_page_num);
         }
         ++hits;
      }
      ++cp;
   }
}
```

```
    if ((hits == 0) && (start_lsn == 0))
       add_card(ssan,buf1, buf2, page);

    else if((hits == 0) && (start_lsn == 1))
       return hits;

    else if (start_lsn){
       cmds[0] = "lsn.exe";
       cmds[1] = "lsn.exe";
       execl(cmds[0],cmds[1], cmds[2], cmds[3],cmds[4],NULL);
    }

    return rc;
}
```

```
/*********************************************************************

 FUNCTION :      list_cards
 CALLED BY:      NONE(Debugging Utility)
 CALLS    :      puts
                 prt_record
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard
 PURPOSE  :      Prints the linked list

*********************************************************************/

static int list_cards(void)
{
  int rc;              /* Indicates an empty list           */

  CARD *tmp;           /* A temporary student information record */

  /*
    Set the tmp student record equal to the head of the linked list
    and check for errors.
  */
  tmp = head;
  if (tmp->next == head){
    puts("List empty");
    ++rc;
  }

  /*
    For each record in the list...print the record
  */
  else
    while (tmp->next != head){
      tmp = tmp->next;
      prt_record(tmp->ssn, tmp->lsn_length,tmp->lsn_name,
              tmp->lsn_page_num);
    }
  retum rc;
}
```

```
/**********************************************************************

 FUNCTION :       error
 CALLED BY:       initialize_linked_list
                  read_file
                  write_file
 CALLS    :       fprintf
                  exit
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard
 PURPOSE  :       Print an error message in the linked list file

**********************************************************************/

void error(char *mesg)
{
  fprintf(stderr, "Error: %s\n", mesg);
  exit(EXIT_FAILURE);
}
```

```
/*********************************************************************

  FUNCTION :        read_file
  CALLED BY:        initialize_linked_list
  CALLS   :         fopen
                    fgets
                    insert_node
                    error
                    strcpy
                    ferror
  MODIFIED :        4/12/90
  PERSON   :        Rick Howard
  PURPOSE  :        Read the student information file into a linked list

*********************************************************************/

static int read_file(char *fname)
{
   char *cp;                /* Used to read data into a buffer */

   FILE *fp;                /* Pointer to a file */

   char line[NBYTES + 1];   /* Used to read data into a buffer */

   int rc = 0;              /* Holds the return value of the function */

   CARD *tmp;               /* Temporary storage of the data template */

   /*
      Open the file for reading.
   */
   fp = fopen(fname, "r");
   if (fp == NULL)
      return 0;

   /*
      Read the data into a buffer.
   */
   while (fgets(line, NBYTES + 1, fp) != NULL){
        /*
            Remove NL.
        */
        cp = line;
        while (*cp != '\n' && *cp != '\0')
            ++cp;
        *cp = '\0';

        /*
            Allocate a node and point to it.
        */
        tmp = insert_node(current);
        if (tmp == NULL)
            error("out of memory");
        current = tmp;

        /*
            Copy data to card structure.
        */
        strcpy(current->ssn, strtok(line, "\t"));
        strcpy(current->lsn_length,strtok(NULL, "\t"));
        strcpy(current->lsn_name, strtok(NULL, "\t"));
        strcpy(current->lsn_page_num, strtok(NULL, "\t"));
    }

    /*
        Close the file.
    */
    fclose(fp);
    if (ferror(fp))
        error("Cannot close data file");

    return rc;
}
```

```
/**********************************************************************

 FUNCTION :      write_file
 CALLED BY:      save_position in the program lsn.exe
 CALLS   :       fopen
                 fprintf
                 puts
                 fclose
                 ferror
                 error
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE :       Write the linked list into the student information file

**********************************************************************/

static int write_file(char *fname)
{
  FILE *fp;              /* File pointer */

  int rc = 0;            /* Holds the return value of the function */

  CARD *tmp;             /* Temporary storage for the data template */

  /*
    Open the file for reading.
  */
  fp = fopen(fname, "w");
  if (fp == NULL){
    fprintf(stderr, "Cannot open %s\n", fname);
    return ++rc;
  }
```

```
    /*
       Write the data into a buffer.
    */
    tmp = head;
    if (tmp->next == head){
       puts("List empty");
       ++rc;
    }
    else
       while (tmp->next != head){
          tmp = tmp->next;
          fprintf(fp, "%s\t%s\t%s\t%s\n", tmp->ssn, tmp->lsn_length,
          tmp->lsn_name, tmp->lsn_page_num);
       }

    /*
       Close the file
    */
    fclose(fp);
    if (ferror(fp))
       error("Cannot close data file");

    return rc;
}
```

```
/**********************************************************************

 FUNCTION :      insert_node
 CALLED BY:      add_card
                 initialize_linked_list
 CALLS   :       malloc
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard
 PURPOSE  :      Inserts a blank node into the linked list

**********************************************************************/

CARD * insert_node(CARD *listp)
{
  CARD *new;

  new = (CARD *) malloc(sizeof(CARD));
  if (new != NULL){
     new->next = listp->next;
     listp->next = new;
  }
  return new;
}
```

```
/*********************************************************************

 FUNCTION :        prt_record
 CALLED BY:        list_cards
 CALLS    :        printf
 MODIFIED :        4/12/90
 PERSON   :        Rick Howard
 PURPOSE  :        Prints one record in the linked list

*********************************************************************/

static void prt_record(char *s1, char *s2, char *s3, char *s4)
{
   printf("%-*s\t%-*s\t%-*s\t%-*s\n", SSNSIZE, s1, LSN_LENGTH_SIZE, s2,
                         LSN_NAME_SIZE, s3, LSN_PAGE_NUM_SIZE,
                         s4);
}
```

```
/*********************************************************************

 FUNCTION :       get_ssn
 CALLED BY:       find_card
 CALLS    :       wopen
                  error_exit
                  wtitle
                  add_shadow
                  winputsf
                  wclose
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard
 PURPOSE  :       Retrieve the user's SSN

*********************************************************************/

static void get_ssn(char buf[10])
{
  /*
    Open a window.
  */
  if(!wopen(5,21,15,58,3,LGREEN|_MAGENTA,LGREEN|_MAGENTA))
       error_exit(1);
  wtitle("[ Enter Social Security Number ]",TCENTER,LGREEN|_MAGENTA);
  add_shadow();

  /*
    Get the user's SSN.
  */
  if(winputsf(buf,"\n\n  Soc Sec Number?  '!R-!"
    "<01234567>##!-!'-'!+!##!-!'-'!+!####")) quit();

  wclose();

}
```

APPENDIX J

THE CODE: FILE "TXTMOD.C"

```
/******************************************************************
                    The Discrete Math Tutor (DMT)
           Thesis Project at the Naval Postgraduate School
             1989-1990 by Keith Calcote and Rick Howard

  FILENAME: txtmod.c

  LIBRARY CALLS:
              exit            Turbo C Lib
              fcloseall       Turbo C Lib
              fopen           Turbo C Lib
              fputs           Turbo C Lib
              fwrite          Turbo C Lib
              getc            Turbo C Lib
              printf          Turbo C Lib
              puts            Turbo C Lib
              strcat          Turbo C Lib
              strcpy          Turbo C Lib
              strtok          Turbo C Lib

  PROGRAM CALLS:
              NONE

  TXTMOD FUNCTIONS:
              pageclr

  COMPLETED:      4/12/90

  PERSONS:        Keith Calcote

  PURPOSE:        Corvert an ASCII file into a format that can be displayed
                  as a lesson in the dmt.exe program

******************************************************************/
```

```
/* header files */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <process.h>
/*----------------------------------------------------------------*/

/* Constants */

#define PAGEL 2000 /* total number of chars per page */
#define NUMLFS 19      /* number of lines per page       */
#define LEN 50  /* number of pages in the tutor   */
#define ZEOF '\x1A'
#define FFEED '\x0C'
#define LFEED '\x0A'
#define CLR printf("\x1B[2J")
/*----------------------------------------------------------------*/

/* globals */

char page[PAGEL];
/*----------------------------------------------------------------*/
```

```
/**********************************************************************

 FUNCTION :      main
 CALLED BY:      NONE
 CALLS    :      See Declarations
 MODIFIED :      4/12/90
 PERSON   :      Keith Calcote
 PURPOSE  :      See Declarations

**********************************************************************/

main(int argc, char *argv[])
{
  int ch;                  /* Temporary character storage          */

  int loop=0, dex=0 ;      /* Counters                     */

  int lincnt = 1 ;         /* Number of lines                  */

  int pgnum = 1 ;          /* Number of pages                  */

  int wordcnt = 0 ;        /* Number of words                   */

  FILE *fptr1 ;            /* Input file pointer             */

  FILE *fptr2 ;            /* Output file pointer             */

  FILE *fptr3 ;            /* Length file pointer             */

  int length[LEN] ;        /* Contains the number of bytes per page     */

  char *chptr ;            /* Temporary character storage pointer      */

  char lenptr[15] ;        /* Temporary file name storage           */

  char output[15] ;        /* Temporary file name storage           */

  /*
    Clears the screen
  */
  CLR ;
```

```
/*
   Error check.
*/
if(argc != 3)
{
   puts("Format is: txtmod inputfile outputfile.") ;
   puts("The outputfile will have the extention .txt .") ;
   puts("A length file will be generated with the same name") ;
   puts("as the output file and will have an extention .len .") ;
   exit(0) ;
}

/*
   Forces the output file to have an extension of ".txt" and forces
   the length file to have the same prefix with the ".len" extension.
*/
chptr = strtok(argv[2],".") ;
strcpy(output,chptr) ;
strcpy(lenptr,chptr) ;
strcat(output,".txt") ;
strcat(lenptr,".len") ;

/*
   Display the name and identification of each file.
*/
printf("Input  file: %s\n",argv[1]) ;
printf("Output file: %s\n",output) ;
printf("Length file: %s\n",lenptr) ;

/*
   Open the files.
*/
if( (fptr1=fopen(argv[1],"rb")) == NULL )
{
   printf("CAN'T OPEN FILE %s ",argv[1]);
   exit(0);
}
if( (fptr2=fopen(output,"wb")) == NULL )
{
   printf("CAN'T OPEN FILE %s ",output);
   exit(0);
}
```

```
if( (fptr3=fopen(lenptr,"wb")) == NULL )
{
  printf("CAN'T OPEN FILE %s",lenptr);
  exit(0);
}

/*
  Counts and stores the number of bytes per page.
*/
pageclr() ;
ch = getc(fptr1) ;
while ( (ch != ZEOF) && (ch != EOF) )
{
  /*
    Prevents pages from being greater than the maximum number
    of lines per page.
  */
  if( lincnt >= NUMLFS )
  {
    lincnt = 1 ;
    fputs(page,fptr2) ;
    pgnum ++ ;
    length[pgnum] = wordcnt ;
    pageclr() ;
    dex = 0 ;

  }

  /*
    Start a new page when a form feed is encountered.
  */
  if (ch == FFEED)
  {
    ch = LFEED ; /* change ch to L/F */
    /*
        Adds line feeds so that each page has the same number of
        lines.
    */
    for(loop = lincnt + 1; loop <= NUMLFS; loop++)
    {
        page[dex] = ch ;
        dex ++ ;
        wordcnt ++ ;
    }
    lincnt = 0 ;
    fputs(page,fptr2) ;
    pgnum ++ ;
    length[pgnum] = wordcnt ;
    pageclr() ;
    dex = 0 ;
}
/*
    Increment the line count and the word count for each line feed.
*/
else
    if ( ch == LFEED )
    {
        lincnt++ ;
        page[dex] = ch ;
        dex ++ ;
        wordcnt ++ ;
    }
/*
    All other characters become part of the page.
*/
else
    {
    page[dex] = ch ;
    dex ++ ;
    wordcnt ++ ;
}
```

```
            /*
                Get the next character.
            */
            ch = getc(fptr1) ;

        }

        /*
            Stores the last page if the last page is not termintated
            with a form feed.
        */
        if(wordcnt != length[pgnum])
        {
            fputs(page,fptr2) ;
            pgnum ++ ;
        }

        /*
            Write the length array to the length file.
        */
        length[pgnum] = wordcnt ;
        length[0] = pgnum - 1 ;
        fwrite(length,sizeof(length),1,fptr3) ;
        fcloseall() ;
    }
```

```
/**********************************************************************

 FUNCTION : pageclr
 CALLED BY: txtmod
 CALLS    : NONE
 MODIFIED : 4/12/90
 PERSON   : Keith Calcote
 PURPOSE  : Puts nulls in page array

**********************************************************************/

static void pageclr()
{
  int loop ;
  for(loop = 0; loop < PAGEL; loop ++)
     page[loop] = '\x00' ;
}

TMOD.C   ***/
```

APPENDIX K

THE CODE: FILE "VENN.C"

```
/**********************************************************************
                    The Discrete Math Tutor (DMT)
             Thesis Project at the Naval Postgraduate School
                1989-1990 by Keith Calcote and Rick Howard

LIBRARY CALLS:
            circle              Turbo C Lib
            cleardevice         Turbo C Lib
            closegraph          Turbo C Lib
            detectgraph         Turbo C Lib
            exit                Turbo C Lib
            floodfill           Turbo C Lib
            getch               Turbo C Lib
            getmaxx             Turbo C Lib
            getmaxy             Turbo C Lib
            initigraph          Turbo C Lib
            line                Turbo C Lib
            moveto              Turbo C Lib
            outtext             Turbo C Lib
            randomize           Turbo C Lib
            rectangle           Turbo C Lib
            setaspectratio      Turbo C Lib
            setbkcolor          Turbo C Lib
            setcolor            Turbo C Lib
            setfillstyle        Turbo C Lib
            settextjustify      Turbo C Lib
            settextstyle        Turbo C Lib

PROGRAM CALLS:
            NONE
```

```
VENN FUNCTIONS:
                correct
                draw
                enter
                incorrect
                info
                reset
                title

COMPLETED:      4/12/90

PERSONS:        Keith Calcote & Rick Howard

PURPOSE:        Provides a the user with a leraning tool that drills the
                relationship between logic expressions and venn diagrams

*******************************************************************/
```

```
/* header files */

#include <graphics.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
/*------------------------------------------------------------------*/

/* function prototypes */

static void correct(void);
static void draw(void);
static void enter(void);
static void incorrect(void);
static void info(void);
static void reset(void);
static void title(void);
/*------------------------------------------------------------------*/
```

```
/* globals */

int driver;                  /* Graphics drive number                      */

int mode ;                   /* Graphics mode number                        */

int n ;                      /* Counter                          */

int left=0;                  /* Left most pixel coordinate                 */

int top=0 ;                  /* Top most pixel coordinate                   */

int xmax,ymax;               /* Max right and bottom coordinate                */

int radius;                  /* Radius of the circles                 */

int c_radius;                /* Corrected radius                     */

int randnum ;                /* Random Number                            */

int header;                  /* Holds the y-coordinate for the header          */

int footer;                  /* Holds the y-coordinate for the footer         */

int gap;                     /* A small number of pixels                   */

int height ;                 /* Verticle distance between the top circle
                                center and the bottom circle center         */

int xposit1, yposit1 ;       /* x/y coordinates for region 1               */

int xposit2, yposit2 ;       /* x/y coordinates for region 2               */

int xposit3, yposit3 ;       /* x/y coordinates for region 3               */

int xposit4, yposit4 ;       /* x/y coordinates for region 4               */

int xposit5, yposit5 ;       /* x/y coordinates for region 5               */

int xposit6, yposit6 ;       /* x/y coordinates for region 6               */

int xposit7, yposit7 ;       /* x/y coordinates for region 7               */
int xposit8, yposit8 ;        /* x/y coordinates for region 8               */

int flag1=0;                  /* Set if region 1 is filled             */

int flag2=0;                  /* Set if region 2 is filled             */

int flag3=0;                  /* Set if region 3 is filled             */

int flag4=0 ;                 /* Set if region 4 is filled             */

int flag5=0;                  /* Set if region 5 is filled             */

int flag6=0;                  /* Set if region 6 is filled             */

int flag7=0;                  /* Set if region 7 is filled             */

int flag8=0 ;                 /* Set if region 8 is filled             */

int sumflag=0 ;               /* Contains total number of regions filles      */

float ratio ;                 /* Used to determine the system aspect ration     */

char ch = 'x';                /* User response                        */
/*-----------------------------------------------------------------*/
```

```
/**********************************************************************

 FUNCTION :       main
 CALLED BY:       NONE
 CALLS    :       See Declarations
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard & Keith Calcote
 PURPOSE  :       See Declarations

**********************************************************************/

main()
{
  while(1)
  {
    /*
      Initialize the graphics mode for the user's screen.
    */
    detectgraph ( &driver , &mode) ;
    initgraph ( &driver, &mode , "c:\\tc") ;
    xmax = getmaxx() ;
    ymax = getmaxy() ;

    /*
      Set the background color to BLUE.
    */
    setbkcolor(1);

    /*
      Calculate the initial screen parameters.
    */
    radius = ymax * 0.238 ;
    header = ymax * 0.15 ;
    footer = ymax * 0.95 ;
    gap = ymax * 0.025 ;
    height = ymax * 0.275 ;
```

```
/*
   Determine the system's aspect ratio.
*/
ratio = (float)ymax/(float)xmax * 10000 * 4 /3 ;
setaspectratio((int) ratio, 10000);
ratio = 10000/ratio ;
c_radius = radius * ratio ;

/*
   Draws the Venn Diagram circles to the screen.
*/
draw() ;

/*
   Pick a random Venn Diagram drawing equation.
*/
randomize() ;
n = 9 ;
randnum = random(n) + 1 ;

/*
   Display the question on the screen.
*/
title() ;

/*
   Display instructions to the screen.
*/
info() ;

/*
   Get the user's response.
*/
ch = getch() ;
```

```
/*
  Based on the user's input, fill each chosen region.
*/
while(ch != '\r')
{
  switch(ch)
  {
  case '1' :
    floodfill(xposit1,yposit1,WHITE) ;
    if(flag1 != 1)
      sumflag++ ;
    flag1 = 1 ;
    break;

  case '2' :
    floodfill(xposit2,yposit2,WHITE) ;
    if(flag2 != 1)
      sumflag++ ;
    flag2 = 1 ;
    break;

  case '3' :
    floodfill(xposit3,yposit3,WHITE) ;
    if(flag3 != 1)
      sumflag++ ;
    flag3 = 1 ;
    break;

  case '4' :
    floodfill(xposit4,yposit4,WHITE) ;
    if(flag4 != 1)
      sumflag++ ;
    flag4 = 1 ;
    break;

  case '5' :
    floodfill(xposit5,yposit5,WHITE) ;
    if(flag5 != 1)
      sumflag++ ;
    flag5 = 1 ;
    break;
    case '6' :
       floodfill(xposit6,yposit6,WHITE) ;
       if(flag6 != 1)
          sumflag++ ;
       flag6 = 1 ;
       break;

    case '7' :
       floodfill(xposit7,yposit7,WHITE) ;
       if(flag7 != 1)
          sumflag++ ;
       flag7 = 1 ;
       break;

    case '8' :
       floodfill(xposit8,yposit8,WHITE) ;
       if(flag8 != 1)
          sumflag++ ;
       flag8 = 1 ;
       break;

    case 'e' :
    case 'E' :
       draw() ;
       reset() ;
       title() ;
       info() ;
       break ;

    case 'q' :
    case 'Q' :
       closegraph() ;
       exit(0) ;
       break;
    default :
       break;

    }/* end switch */

    ch = getch() ;

 }/* end while */
```

```
/*
   Determines if the user's answer is correct.
*/
switch(randnum)
{
/*
   A' intersect B' intersect C'
*/
case 1 :
   if(sumflag == 1 && flag8 == 1)
   {
      correct() ;
      reset() ;
   }
   else
      {
      incorrect() ;
      getch() ;
      draw() ;
      title() ;
      floodfill(xposit8,yposit8,WHITE) ;
      enter() ;
      reset() ;
   }
   break;
```

```
/*
  A intersect B intersect C
*/
case 2 :
  if(sumflag == 1 && flag7 == 1)
  {
    correct() ;
    reset() ;
  }
  else
    {
    incorrect() ;
    getch() ;
    draw() ;
    title() ;
    floodfill(xposit7,yposit7,WHITE) ;
    enter() ;
    reset() ;
  }
  break;

/*
  A' intersect B intersect C
*/
case 3 :
  if(sumflag == 1 && flag6 == 1)
  {
    correct() ;
    reset() ;
  }
  else
    {
    incorrect() ;
    getch() ;
    draw() ;
    title() ;
    floodfill(xposit6,yposit6,WHITE) ;
    enter() ;
    reset() ;
  }
  break;
```

```
/*
    A intersect B intersect C'
*/
case 4 :
    if(sumflag == 1 && flag5 == 1)
    {
        correct() ;
        reset() ;
    }
    else
        {
        incorrect() ;
        getch() ;
        draw() ;
        title() ;
        floodfill(xposit5,yposit5,WHITE) ;
        enter() ;
        reset() ;
    }
    break;

/*
    A intersect B' intersect C
*/
case 5 :
    if(sumflag == 1 && flag4 == 1)
    {
        correct() ;
        reset() ;
    }
    else
        {
        incorrect() ;
        getch() ;
        draw() ;
        title() ;
        floodfill(xposit4,yposit4,WHITE) ;
        enter() ;
        reset() ;
    }
    break;
```

```
/*
  C intersect (B union A)'
*/
case 6 :
  if(sumflag == 1 && flag3 == 1)
  {
    correct() ;
    reset() ;
  }
  else
    {
    incorrect() ;
    getch() ;
    draw() ;
    title():
    floodfill(xposit3,yposit3,WHITE) ;
    enter() ;
    reset() ;
  }
  break;

/*
  B intersect (C union A)'
*/
case 7 :
  if(sumflag == 1 && flag2 == 1)
  {
    correct() ;
    reset() ;
  }
  else
    {
    incorrect() ;
    getch() ;
    draw() ;
    title() ;
    floodfill(xposit2,yposit2,WHITE) ;
    enter() ;
    reset() ;
  }
  break;
```

```
/*
   A intersect (B union C)'
*/
case 8 :
   if(sumflag == 1 && flag1 == 1)
   {
      correct() ;
      reset() ;
   }
   else
      {
      incorrect() ;
      getch() ;
      draw() ;
      title() ;
      floodfill(xposit1,yposit1,WHITE) ;
      enter() ;
      reset() ;
   }
   break;
```

```
    /*
      A union B union C
    */
    case 9 :
      if(sumflag == 7 && flag8 == 0)
      {
        correct() ;
        reset() ;
      }
      else
        {
        incorrect() ;
        getch() ;
        draw() ;
        title() ;
        floodfill(xposit1,yposit1,WHITE) ;
        floodfill(xposit2,yposit2,WHITE) ;
        floodfill(xposit3,yposit3,WHITE) ;
        floodfill(xposit4,yposit4,WHITE) ;
        floodfill(xposit5,yposit5,WHITE) ;
        floodfill(xposit6,yposit6,WHITE) ;
        floodfill(xposit7,yposit7,WHITE) ;
        enter() ;
        reset();
      }
      break;

    }

    getch() ;
  }
}
```

```
/*************************************************************************

 FUNCTION :      draw
 CALLED BY:      venn
 CALLS    :      cleardevice
                 settextstyle
                 setcolor
                 rectangle
                 line
                 moveto
                 outtext
                 circle
                 setfillstyle
 MODIFIED :      4/12/90
 PERSON   :      Keith Calcote
 PURPOSE  :      Draws the outline and circles to the screen.

*************************************************************************/

static void draw(void)
{
  /*
    Initializes graphics text font.
  */
  cleardevice() ;
  settextstyle(DEFAULT_FONT,HORIZ_DIR,1) ;
  setcolor(WHITE) ;

  /*
    Draws the basic frame.
  */
  rectangle(left, top, xmax, ymax) ;
  line(0,header,xmax,header) ;
  line(0,footer,xmax,footer) ;

  /*
    Prints "#1" in region 1.
  */
  xposit1 = xmax /2 ;
  yposit1 = radius + header + gap ;
  moveto(xposit1+1,yposit1-gap) ;
  outtext("1") ;
```

```
/*
   Prints "#2" in region 2.
*/
xposit2 = xposit1 + radius * ratio * 2/3 ;
yposit2 = yposit1 + height ;
moveto(xposit2+1,yposit2) ;
outtext("2") ;

/*
   Prints "#3" in region 3.
*/
xposit3 = xposit1 - radius * ratio * 2/3 ;
yposit3 = yposit2 ,
moveto(xposit3+1,yposit3) ;
outtext("3") ;

/*
   Prints "#4" in region 4.
*/
xposit4 = xposit1 - radius * ratio /3 ;
yposit4 = yposit1 + radius /sqrt(3) ;
moveto(xposit4+1,yposit4) ;
outtext("4") ;

/*
   Prints "#5" in region 5.
*/
xposit5 = xposit1 + radius * ratio /3 ;
yposit5 = yposit4 ;
moveto(xposit5+1,yposit5) ;
outtext("5") ;

/*
   Prints "#6" in region 6.
*/
xposit6 = xposit1 ;
yposit6 = yposit2 ;
moveto(xposit6+1,yposit6) ;
outtext("6") ;
```

```
    /*
       Prints "#7" in region 7.
    */
    xposit7 = xposit1 ;
    yposit7 = yposit1 + radius * 2/3 ;
    moveto(xposit7+1,yposit7) ;
    outtext("7") ;

    /*
       Prints "#8" in region 8.
    */
    xposit8 = 2 * gap * ratio ;
    yposit8 = header + 2 * gap ;
    moveto(xposit8+1,yposit8) ;
    outtext("8") ;

    /*
       Draws the three circles.
    */
    circle(xposit1,yposit1,c_radius) ;
    circle(xposit2,yposit2,c_radius) ;
    circle(xposit3,yposit3,c_radius) ;

    /*
       Draws the letters A, B, & C in the three circles.
    */
    settextstyle(DEFAULT_FONT,HORIZ_DIR,2) ;
    moveto(xposit1 , yposit1-radius/2) ;
    outtext("A") ;
    moveto(xposit2 + radius/2 , yposit2+ 2*gap) ;
    outtext("B") ;
    moveto(xposit3 - radius/2, yposit3 + 2*gap) ;
    outtext("C") ;

    /*
       Resets the text style to default.
    */
    settextstyle(DEFAULT_FONT,HORIZ_DIR,1) ;
    setfillstyle(LTSLASH_FILL,WHITE) ;

}
```

```
/*************************************************************************

 FUNCTION :       reset
 CALLED BY:       venn
 CALLS    :       NONE
 MODIFIED :       4/12/90
 PERSON   :       Keith Calcote
 PURPOSE  :       Resets all flags to zero.

*************************************************************************/

static void reset(void)
{
   flag1 = 0 ;
   flag2 = 0 ;
   flag3 = 0 ;
   flag4 = 0 ;
   flag5 = 0 ;
   flag6 = 0 ;
   flag7 = 0 ;
   flag8 = 0 ;
   sumflag = 0 ;
}
```

```
/*********************************************************************

 FUNCTION :      title
 CALLED BY:      venn
 CALLS    :      moveto
                 settextjustify
                 settextstyle
                 outtext
 MODIFIED :      4/12/90
 PERSON   :      Keith Calcote
 PURPOSE  :      Prints the title of the associated Venn Diagram.

*********************************************************************/

title()
{
  /*
    Moves the cursor position to the top of the screen and sets
    the text justification and style.
  */
  moveto(xmax /2, header /2) ;
  settextjustify(CENTER_TEXT,CENTER_TEXT) ;
  settextstyle(DEFAULT_FONT,HORIZ_DIR,2) ;

  /*
    Displays the diagram name across the top of the sceen.
  */
  switch(randnum)
  {
  case 1 :
    outtext("A' intersect B' intersect C' ?") ;
    break;

  case 2 :
    outtext("A intersect B intersect C ?") ;
    break;

  case 3 :
    outtext("A' intersect B intersect C ?") ;
    break;
  case 4 :
    outtext("A intersect B intersect C' ?") ;
    break;

  case 5 :
    outtext("A intersect B' intersect C ?") ;
    break;

  case 6 :
    outtext("C intersect (B union A)' ?") ;
    break;

  case 7 :
    outtext("B intersect (C union A)' ?") ;
    break;

  case 8 :
    outtext("A intersect (B union C)' ?") ;
    break;

  case 9 :
    outtext("A union B union C ?") ;
    break;

  }

  /*
    Resets the text justification and style to default.
  */
  settextjustify(LEFT_TEXT,TOP_TEXT) ;
  settextstyle(DEFAULT_FONT,HORIZ_DIR,1) ;

}
```

```
/********************************************************************

 FUNCTION :      info
 CALLED BY:      venn
 CALLS   :       moveto
                 outtext
 MODIFIED :      4/12/90
 PERSON  :       Keith Calcote
 PURPOSE :       Displays text across the bottom of the screen.

********************************************************************/

static void info(void)
{
  moveto(gap,footer+gap/2) ;
  outtext("push 1-8 to fill, q = quit, e = erase") ;
  moveto(xmax *3/5,footer+gap/2) ;
  outtext("enter to continue            " ) ;
}

/********************************************************************

 FUNCTION :      correct
 CALLED BY:      venn
 CALLS   :       moveto
                 outtext
 MODIFIED :      4/12/90
 PERSON  :       Keith Calcote
 PURPOSE :       Displays "CORRECT" at the bottom right hand side of the
                 screen.

********************************************************************/

correct()
{
  moveto(xmax *4/5,footer-2*gap) ;
  outtext("CORRECT") ;
}
```

```
/*********************************************************************

  FUNCTION :      incorrect
  CALLED BY:      venn
  CALLS   :       moveto
                  outtext
  MODIFIED :      4/12/90
  PERSON  :       Keith Calcote
  PURPOSE  :      Displays "INCORRECT" at the bottom right hand side of the
                  screen.

*********************************************************************/

incorrect()
{
  moveto(xmax *4/5,footer-2*gap) ;
  outtext("INCORRECT") ;
}

/*********************************************************************

  FUNCTION :      enter
  CALLED BY:      venn
  CALLS   :       moveto
                  outtext
  MODIFIED :      4/12/90
  PERSON  :       Keith Calcote
  PURPOSE  :      Displays text at the bottom of the screen.

*********************************************************************/

enter()
{
  moveto(gap,footer+gap/2) ;
  outtext("CORRECT SOLUTION IS PRESENTED") ;
  moveto(xmax *3/5,footer+gap/2) ;
  outtext("enter to continue          " ) ;

}
```

APPENDIX L

THE CODE: FILE "PRINT.C"

```
/**********************************************************************
                    The Discrete Math Tutor (DMT)
             Thesis Project at the Naval Postgraduate School
              1989-1990 by Keith Calcote and Rick Howard

 LIBRARY CALLS:
                 exit
                 fclose
                 fgets
                 fopen
                 fputs
                 printf
 PROGRAM CALLS:
                 NONE

 PRINT FUNCTIONS:
                 NONE

 COMPLETED:      4/12/90

 PERSONS:        Rick Howard

 PURPOSE:        Sends a file to the printer

**********************************************************************/

/* header files */

#include <stdio.h>
/*----------------------------------------------------------------*/
```

```
/************************************************************************

  FUNCTION :       main
  CALLED BY:       NONE
  CALLS    :       See Declarations
  MODIFIED :       4/12/90
  PERSON   :       Rick Howard
  PURPOSE  :       See Declarations

************************************************************************/

main(argc,argv)
int argc;
char *argv[];
{
  FILE *fptr1;        /* Pointer to the file to be printed */

  FILE *fptr2;        /* Pointer to the printer device    */

  FILE *fptr3;        /* Pointer to a NULL file           */

  char string[81];    /* Array that holds each page of the file */

  /*
     Error checking.
  */
  if(argc != 2)
  {
     printf("Format: C>print filename");
     exit();
  }
  if(( fptr1=fopen(argv[1],"r")) == NULL)
  {
     printf("Can't open file %s.", argv[1]);
     exit();
  }
  if(( fptr2=fopen("prn", "w")) == NULL)
  {
     printf("Can't access printer.");
     exit();
  }
```

```
    if(( fptr3=fopen("nothing.def","r")) == NULL)
    {
      printf("Can't open nothing.def");
      exit();
    }


    /*
      Send one page at a time to the printer.
    */
    while (fgets(string,80,fptr1) != NULL)
      fputs(string.fptr2);

    /*
      Clear the buffer.
    */
    while (fgets(string,80,fptr3) != NULL)
      fputs(string.fptr2);


    fclose(fptr1);
    fclose(fptr2);
    fclose(fptr3);
}
```

APPENDIX M

THE CODE: FILE "EXAM.C"

```
/*********************************************************************
                 The Discrete Math Tutor (DMT)
          Thesis Project at the Naval Postgraduate School
            1989-1990 by Keith Calcote and Rick Howard

 LIBRARY CALLS:
           error_exit       DMT Utilities
           exit             Turbo Lib
           fcloseall        Turbo Lib
           fread            Turbo Lib
           fseek            Turbo Lib
           getch            Turbo Lib
           printf           Turbo Lib
           random           Turbo Lib
           randomize        Turbo Lib
           set_video        DMT Utilities
           strcat           Turbo Lib
           strcpy           Turbo Lib
           strtok           Turbo Lib
           waitkey          CXL Lib
           whelpcat         CXL Lib
           whelpdef         CXL Lib
           wopen            CXL Lib
           wprintf          CXL Lib
           wputsw           CXL Lib
           wtextattr        CXL Lib

 PROGRAM CALLS:
           NONE
 EXAM FUNCTIONS:
                 create_length_files
                 error_check
                 explanations
                 error_msg
                 get_answer
                 get_question
                 highlight_correct_answer
                 initialize
                 open_window
                 pageclr
                 quit
                 upr_lwr_case

 COMPLETED: 4/12/90

 PERSONS: Rick Howard & Keith Calcote

 PURPOSE: Present the user with an exam for any generic lesson.

*****************************************************************************/

/* header files */

#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include <conio.h>
#include <time.h>
#include <string.h>
#include <math.h>
#include "d:\cxl\cxlwin.h"
#include "d:\cxl\cxlkey.h"
#include "d:\cxl\cxlvid.h"
#include "d:\tc\thesis\video.h"
#include "d:\tc\thesis\help.h"
/*-----------------------------------------------------------------*/
```

```
/* function prototypes */

static void error_check(int argc);
static void error_msg(int msg_num, char *string, int integer);
static void create_length_files(char *argv[]);
static void open_window(void);
static void initialize(char *argv[]);
static void get_question(void);
static void get_answer(void);
static void upr_lwr_case(void);
static void check_answer(void);
static void highlight_correct_answer(void);
static void explanations(void);
static void results(void);
static void pageclr(void);
static void quit(void);
/*-----------------------------------------------------------------*/

/* constatnts */

#define ROW 25
#define COL 1
#define LEN 100
#define PAGEL 2000
#define CLR wcclear(WHITE|_CYAN)
#define BOTTOM_LEFT wgotoxy(15,0)
/*-----------------------------------------------------------------*/
```

```
/* globals */

char page[PAGEL] ;          /* Holds the words on each exam page */

WINDOW w,w1;                /* Window handles                    */

int dummy_int;              /* Place holder for the error_msg function */

int qlength[LEN];           /* Contains the number of bytes per page
                                in the question file             */

int elength[LEN];           /* Contains the number of bytes per page
                               in the explanation file           */

int pflag;                  /* Set after the first pass thru the
                               question file.  Allows the program to
                               strip off the "@" characters      */

int loop;                   /* Counter used to clear the page buffer  */

int used_stack[LEN] ;       /* Array that holds the exam questions
                               already presented to the user     */

int ch;                     /* Used to get the user's response   */

int recno;                  /* Desired page number for the question
                               and explanation file              */

int adjustment;             /* Used to accept both upper and lower
                               case input from the user          */

int num_quest;              /* The  number of exam questions
                               desired by the user               */

int dex;                    /* Counter used to annotate the number of
                               questions presented to the user   */

int n ;                     /* Counter                           */

int num_correct = 0 ;       /* Used to keep track of the number of
                               correct answers given by the user */
```

```
int num_incorrect = 0 ;        /* Used to keep track of the number of
                                  incorrect answers given by the user    */

long int q_offset,e_offset ;   /* Used to point to a desired page in the  text */

float grade ;                  /* Percentage based upon the user's
                                  number of correct answers divided by
                                  the total number of exam questions     */

FILE *fptr1 ;                  /* Pointer to the file of questions       */

FILE *fptr2 ;                  /* Pointer to the question length file    */

FILE *fptr3 ;                  /* Pointer to the file of explanations    */

FILE *fptr4 ;                  /* Pointer to the explanation length file  */

char *dummy_string;            /* Place holder for the error_msg function */

char *chptr;                   /* Holds the value of returned by the function
                                  strtok()  */

char quest_len[15],
    expl_len[15];              /* Holds the file name that contains the
                                  length array of the associated file    */

char quest_txt[15],
    expl_txt[15] ;             /* Holds the file name that contains the
                                  text for the associated file        */

char answer[500];              /* Holds the answers to the questions     */

char your_ans;                 /* Used to hold the user's response       */

char ch1 ;                     /* Used to hold the user's response       */
```

```
/*********************************************************************

 FUNCTION :      main
 CALLED BY:      NONE
 CALLS   :       See Declarations
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard & Keith Calcote
 PURPOSE :        See Declarations

*********************************************************************/

main(int argc, char *argv[])
{
  create_length_files(argv);
  error_check(argc);
  open_window();
  initialize(argv);
  for(dex = 0; dex < num_quest; dex ++)
  {
    get_question();
    get_answer();
    upr_lwr_case();
    check_answer();
    highlight_correct_answer();
    explanations();
  }
  results();
  fcloseall();
}
```

```
/***********************************************************************

 FUNCTION :      error_check
 CALLED BY:      exam
 CALLS   :       error_msg
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard & Keith Calcote
 PURPOSE  :      Determines any initialization errors and displays the
                 the appropriate error message.

***********************************************************************/

static void error_check(int argc)
{
  /*
    This program must have four arguments.
  */
  if (argc != 4)
    error_msg(1,dummy_string,dummy_int);

  /*
    Errors in opening the needed files.
  */
  if ((fptr1=fopen(quest_txt,"rb")) == NULL)
    error_msg(2,quest_txt,dummy_int);
  if ((fptr2=fopen(quest_len,"rb")) == NULL)
    error_msg(2,quest_len,dummy_int);
  if ((fptr3=fopen(expl_txt,"rb")) == NULL)
    error_msg(2,expl_txt,dummy_int);
  if ((fptr4=fopen(expl_len,"rb")) == NULL)
    error_msg(2,expl_len,dummy_int);
}
```

```
/**********************************************************************

 FUNCTION :      create_length_files
 CALLED BY:      exam
 CALLS    :      strtok
                 strcpy
                 strcat
 MODIFIED :      4/12/90
 PERSON   :       Rick Howard & Keith Calcote
 PURPOSE  :      Creates the file neames: quest_len, quest_txt, expl_len and
                 expl_txt from the command line.

**********************************************************************/

static void create_length_files(char *argv[])
{
   chptr = strtok(argv[2],".") ;
   strcpy(quest_txt,chptr) ;
   strcpy(quest_len,chptr) ;
   strcat(quest_len,".len") ;
   strcat(quest_txt,".txt") ;

   chptr = strtok(argv[3],".") ;
   strcpy(expl_txt,chptr) ;
   strcpy(expl_len,chptr) ;
   strcat(expl_len,".len") ;
   strcat(expl_txt,".txt") ;
}
```

```
/*********************************************************************

 FUNCTION :      open_window
 CALLED BY:      exam
 CALLS    :      set_video
                 wopen
                 error_exit
                 whelpdef
                 whelpcat
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard
 PURPOSE  :      Opens a window on the screen for the exam

*********************************************************************/

static void open_window(void)
{
  /*
    Check for mono, CGA or EGA screen.
  */
  set_video();

  /*
    Open a window to display the exam.
  */
  if((w=wopen(2,0,23,79,3,WHITE|_CYAN,WHITE|_CYAN))==0)
    wprintf("error_exit(1);");

  /*
    Define the help screen attributes.
  */
   whelpdef("DMT.HLP", 0x2368, BLACK|_LGREY,BLACK|_LGREY,
                                 LBLUE|_LGREY, LRED|_LGREY, 0);

  /*
    Set the current help screen.
  */
  whelpcat(H_TRUTH_TABLE_PROBLEM_SOLVER);
}
```

```
/************************************************************************

 FUNCTION :       initialize
 CALLED BY:       exam
 CALLS    :       fread
                  atoi
                  randomize
                  error_msg
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard & Keith Calcote
 PURPOSE  :       Sets the program up with user supplied paramenters and
                  provides initial error checking.

************************************************************************/

static void initialize(char *argv[])
{
   fread(qlength.sizeof(qlength),1,fptr2) ;
   fread(elength.sizeof(elength),1,fptr4) ;

   /*
      Set the number of questions desired by the user.
   */
   num_quest = atoi(argv[1]) ;
   randomize() ;

   /*
      The explanation file must have the same number of explanations as
      the question file has questions or there exists an error.
   */
   if(qlength[0] != elength[0])
      error_msg(3,dummy_string, dummy_int);

   /*
      The number of required questions must be less than the number of
      questions available.
   */
   if(num_quest > qlength[0])
      error_msg(4,dummy_string,qlength[0]);
```

```
    /*
      The total number of questions top be presented is placed on top
      of a stack.
    */
    for(dex = 0; dex < LEN; dex ++ )
      used_stack[dex] = 0 ;
  }
```

```
/************************************************************************

 FUNCTION :      get_question
 CALLED BY:      exam
 CALLS   :       random
                 fseek
                 error_msg
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard & Keith Calcote
 PURPOSE :       Retrieve an exam question from the exam file

************************************************************************/

static void get_question(void)
{
  /*
     Clear the window.
  */
  CLR ;

  /*
     Choose a random exam question.
  */
  recno = random(qlength[0]) + 1 ;

  /*
     If the question has already been answered, then increment the
     question number and check again.
  */
  for(n = 0; n < dex; n ++ )
  {
     if( recno == used_stack[n])
     {
        if(recno == qlength[0])
           recno = 1 ;
        else
           recno ++ ;
        n = -1 ;
     }
  }
```

```
    /*
       Places the selected question on the used stack and sets the file
       pointer to the desired question in the question file.
    */
    used_stack[dex] = recno ;
    q_offset = qlength[recno] ;
    if( fseek(fptr1,q_offset,0) != 0)
       error_msg(5,dummy_string,dummy_int);
}
```

```
/*******************************************************************

 FUNCTION :      get_answer
 CALLED BY:      exam
 CALLS   :       pageclr
                 fread
                 printf
                 strtok
                 wprintf
                 strcpy
                 getch
                 quit
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard & Keith Calcote
 PURPOSE  :      Displays the question to the user and retrieves the corect
                 answer.

*******************************************************************/

static void get_answer(void)
{
  /*
    Clear the page buffer.
  */
  pageclr();

  /*
    Sets the pointer, chptr, to the value returned by strtok().
  */
  fread(page,qlength[recno+1]-qlength[recno],1,fptr1) ;
  printf("\n") ;
  pflag = 0 ;
  chptr = strtok(page,"@") ;
```

```
    /*
      Strips off the @ characters and displays the question.
    */
    while(chptr != NULL)
    {
      wprintf("%s",chptr) ;
      if(pflag == 0)
      {
        pflag = 1 ;
        chptr = strtok(NULL,"@") ;
        strcpy(answer,chptr) ;
        strcpy(&answer[1],NULL) ;
      }
      else
        {
        chptr = strtok(NULL,"@") ;
      }
    }

    /*
      Retrieve the user's choice.
    */
    your_ans = getch() ;

    /*
      Terminate the program if the user desires.
    */
    if (your_ans == 'Q' || your_ans == 'q')
      quit();

    /*
      Clear the window.
    */
    CLR ;
}
```

```
/**********************************************************************

 FUNCTION :      upr_lwr_case
 CALLED BY:      exam
 CALLS   :       NONE
 MODIFIED :      4/12/90
 PERSON  :       Keith Calcote
 PURPOSE :       Checks for user input in either upper or lower case and
                 makes the proper adjustment.

**********************************************************************/

static void upr_lwr_case(void)
{
  /*
    Sets the adjustment to 32 if the character that corresponds to the
    answer is in lower case.
  */
  if( (int)answer[0] >= 97 && (int)answer[0] <=122 )
    adjustment = 32 ;

  /*
    Sets the adjustment to -32 if the character that corresponds to the
    answer is in upper case.
  */
  else if( (int)answer[0] >= 65 && (int)answer[0] <= 90 )
    adjustment = -32 ;

  /*
     Sets the adjustment to 0 if the character  that corresponds to the
     answer is not a letter.
  */
  else
    adjustment = 0 ;

}
```

```
/*********************************************************************

 FUNCTION :      check_answer
 CALLED BY:      exam
 CALLS    :      wprintf
                 pageclr
 MODIFIED :      4/12/90
 PERSON   :      Keith Calcote & Rick Howard
 PURPOSE  :      Checks the user's response for correctness.

*********************************************************************/

static void check_answer(void)
{

  if(your_ans == answer[0] ||
     ((int)your_ans + adjustment) == (int)answer[0] )

  {

     wprintf("Your answer %c was CORRECT.\n",your_ans) ;
     num_correct ++ ;
  }
  else
     {
     wprintf("Your answer %c was INCORRECT.\n",your_ans) ;
     num_incorrect ++ ;
  }


  pageclr() ;

}
```

```
/*************************************************************************

  FUNCTION :       highlight_correct_answer
  CALLED BY:       exam
  CALLS    :       fseek
                   error_msg
                   fread
                   strtok
                   wprintf
                   wtextattr
  MODIFIED :       4/12/90
  PERSON   :       Keith Calcote & Rick Howard
  PURPOSE  :       Highlights the correct answer on the screen.

*************************************************************************/

static void highlight_correct_answer(void)
{
  /*
     Reads the question into the character array page.
  */
  if( fseek(fptr1,q_offset,0) != 0)
     error_msg(5,dummy_string,dummy_int);
  fread(page,qlength[recno+1]-qlength[recno],1,fptr1) ;

  /*
     Sets the pointer to the "@" in the question.
  */
  pflag = 0 ;
  chptr = strtok(page,"@") ;
  /*
    Points to the highlighted question.
  */
  while(chptr != NULL)
  {
    wprintf("%s",chptr) ;
    if(pflag == 0)
    {
      wtextattr(LCYAN|_GREEN|BLINK);
      pflag = 1 ;
    }
    else
      wtextattr(WHITE|_CYAN);
    chptr = strtok(NULL,"@") ;
  }
}
```

```
/*********************************************************************

 FUNCTION :        explanations
 CALLED BY:        exam
 CALLS   :         wprintf
                   getch
                   quit
                   fseek
                   error_msg
                   pageclr
                   fread
                   wputsw
 MODIFIED :        4/12/90
 PERSON  :         Keith Calcote & Rick Howard
 PURPOSE  :        Provides an explanation to the user concerning the current
                   exam question if desired.

*********************************************************************/

static void explanations(void)
{
  /*
    Position the cursor at the bottom left of the window.
  */
  BOTTOM_LEFT;
  wprintf("E for explanation, enter to continue") ;

  /*
    Get the user's response.
  */
  ch1 = getch() ;

  if (ch1 == 'Q' || ch1 == 'q')
    quit();
  CLR;
  if(ch1 == 'E' || ch1 == 'e')
  {
    /*
      Sets the offset for the explanation page.
    */
    e_offset = elength[recno] ;
```

```
      /*
         Moves the file pointer to the desired offset.
      */
      if( fseek(fptr3,e_offset,0) != 0)
         error_msg(5,dummy_string,dummy_int);

      /*
         Clear the page buffer.
      */
      pageclr();

      /*
         Reads the explanationinto the character array page.
      */
      fread(page,elength[recno+1]-elength[recno],1,fptr3) ;

      /*
         Displays the explanation on the screen.
      */
      wprintf('\n");
      wputsw(page);
      BOTTOM_LEFT;
      wprintf("Push enter to continue") ;
      getch() ;

   }
}
```

```
/*********************************************************************

 FUNCTION :      results
 CALLED BY:      exam
 CALLS    :      wprintf
                 getch
 MODIFIED :      4/12/90
 PERSON   :      Keith Calcote & Rick Howard
 PURPOSE  :      Shows the user his performance on the exam.

*********************************************************************/

static void results(void)
{
   CLR ;
   wprintf("You answered %d correctly\n",num_correct) ;
   wprintf("You answered %d incorrectly\n\n",num_incorrect) ;
   grade = (float)num_correct/(float)num_quest*100.0 ;
   wprintf("GRADE  %3.1f%",grade) ;
   getch() ;

}
```

```
/************************************************************************

 FUNCTION :       pageclr
 CALLED BY:       get_answer
                  check_answer
                  explanations
 CALLS    :       NONE
 MODIFIED :       4/12/90
 PERSON   :       Keith Calcote
 PURPOSE  :       Clears the page buffer

************************************************************************/

static void pageclr(void)
{
   int loop ;
   for(loop = 0; loop < PAGEL; loop ++)

      page[loop] = '\x00' ;
}
```

```
/**********************************************************************

 FUNCTION :       error_msg
 CALLED BY:       error_check
                  initialize
                  get_question
                  highlight_correct_answer
                  explanations
 CALLS   :        wprintf
                  waitkey
 MODIFIED :       4/12/90
 PERSON  :        Rick Howard
 PURPOSE :        Displays the appropriate error message for a known problem.

**********************************************************************/

static void error_msg(int msg_num, char *string, int integer)
{
  switch (msg_num)
  {
  case 1:
    wprintf("Format is:  Q_&_A
             Number_of_questions
             question_file
             explanation_file");
    break;
  case 2:
    wprintf("Can't open %s", string);
    break;
  case 3:
    wprintf("There must be an explanation file for each question");
    break;
  case 4:
    wprintf("You can request at most %d questions", integer);
    break;
  case 5:
    wprintf("Can not move Pointer there!");
    break;
  default:
    break;
  }
```

```
    /*
       Wait for the user's response and quit.
    */
    waitkey();
    exit(0);
}
```

```
/***********************************************************************

 FUNCTION :     quit
 CALLED BY:     get_answer
                explanations
 CALLS    :     exit
 MODIFIED :     4/12/90
 PERSON   :     Rick Howard
 PURPOSE  :     Terminates the program

***********************************************************************/

static void quit(void)
{
  exit(0);
}
```

APPENDIX N

THE CODE: FILE "VENNINFO.C"

```
/**********************************************************************
                 The Discrete Math Tutor (DMT)
          Thesis Project at the Naval Postgraduate School
            1989-1990 by Keith Calcote and Rick Howard

 LIBRARY CALLS:
              circle              Turbo C Lib
              cleardevice         Turbo C Lib
              clrscr              Turbo C Lib
              closegraph          Turbo C Lib
              detectgraph         Turbo C Lib
              exit                Turbo C Lib
              floodfill           Turbo C Lib
              getch               Turbo C Lib
              getmaxx             Turbo C Lib
              getmaxy             Turbo C Lib
              gettext             Turbo C Lib
              initigraph          Turbo C Lib
              line                Turbo C Lib
              moveto              Turbo C Lib
              outtext             Turbo C Lib
              puts                Turbo C Lib
              puttext             Turbo C Lib
              randomize           Turbo C Lib
              rectangle           Turbo C Lib
              setaspectratio      Turbo C Lib
              setbkcolor          Turbo C Lib
              setcolor            Turbo C Lib
              setfillstyle        Turbo C Lib
              settextjustify      Turbo C Lib
              settextstyle        Turbo C Lib
              window              Turbo C Lib

 PROGRAM CALLS:
              NONE
```

```
VENNINFO FUNCTIONS:
                draw
                title

COMPLETED: 4/12/90

PERSONS: Keith Calcote

PURPOSE:  Provides a the user with a leraning tool that drills the
               relationship between logic expressions and venn diagrams

****************************************************************************/
```

```
/* header files */

#include <graphics.h>
#include <stdlib.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
/*-----------------------------------------------------------------*/

/* Constants */

#define NUMLIST 16
#define FONT 0
#define CHSIZE 5
/*-----------------------------------------------------------------*/
```

```
/* Globals */

int driver;                     /*  Graphics driver number        */

int mode ;                      /*  Graphics mode number          */

int n ;                         /*  Counter                  */

int left=0,top=0 ;              /*  Left hand and top most positions */

int xmax,ymax;                  /*  Holds the max number of pixels */

int radius;                     /*  Max desired radius of circles  */

int c_radius;                   /*  Radius corrected for aspect ratio  */

int header,footer,gap,height ;  /*  Screen pixel locations        */

                                /*  x,y coordinate position in
                                    pixels that correspond to
                                    regions in the Venn Diagram    */
int xposit1, yposit1 ;
int xposit2, yposit2 ;
int xposit3, yposit3 ;
int xposit4, yposit4 ;
int xposit5, yposit5 ;
int xposit6, yposit6 ;
int xposit7, yposit7 ;
int xposit8, yposit8 ;

int textbuff[4000] ;            /*  Menu Storage             */

float ratio ;                   /*  Compensation for non-square pixels  */

char ch ;                       /*  Holds user selection        */
/*-------------------------------------------------------------------*/
```

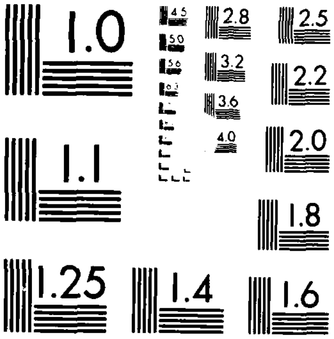
```
/**********************************************************************

 FUNCTION :        main
 CALLED BY:        NONE
 CALLS    :        See Declarations
 MODIFIED :        4/12/90
 PERSON   :        Rick Howard
 PURPOSE  :        See Declarations

*********************************************************************/

main()
{
  /*
    Possible venn diagrams for the user to view
  */
  char list[NUMLIST][80] =
    {
    " a. A' intersect B' intersect C'",
    " b. A  intersect B  intersect C ",
    " c. A' intersect B  intersect C ",
    " d. A  intersect B  intersect C'",
    " e. A  intersect B' intersect C ",
    " f. C  intersect (A union B)' ",
    " g. B  intersect (C union A)' ",
    " h. A  intersect (B union C)' " ,
    " i. A  union B  union C  ",
    " j. A' intersect B' ",
    " k. B' intersect C' ",
    " l. A  union B' ",
    " m. B  union C' ",
    " n. A  union C' ",
    " o. B  union C  ",
    " q. quit "
  };
```

```
/*
   Default an italic type of graphics.
*/
detectgraph ( &driver , &mode) ;
initgraph ( &driver, &mode ,NULL) ;
xmax = getmaxx() ;
ymax = getmaxy() ;

/*
   Initial screen setup.
*/
settextstyle(FONT+1,0,CHSIZE) ;
moveto(xmax/2,ymax/2-75) ;
settextjustify(1,1) ;
outtext("Welcome to ") ;
moveto(xmax/2,ymax/2+20) ;
settextstyle(FONT+1,0,4) ;
outtext("Venn Diagram Information") ;
moveto(xmax/2,ymax-20) ;
settextstyle(FONT,0,1) ;
outtext("Q = quit, Any other key begins") ;

/*
   Get user selection from the initial screen.
*/
ch = getch() ;
closegraph() ;
if (ch == 'q' || ch == 'Q' )
{
   clrscr() ;
   exit(0) ;
}

/*
   Print to screen menu list
*/
window(0,0,80,25) ;
clrscr() ;
for(n=0; n<NUMLIST; n++)
   puts(&list[n][0]) ;
gettext(0,0,80.25,textbuff) ;
ch = getch() ;
```

AD-A232 693 USER INTERFACE TO AN ICAI SYSTEM THAT TEACHES DISCRETE 4/4
MATH(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA
R K CALCOTE ET AL. JUN 90 XN-NPS

UNCLASSIFIED NL

END
FILMED
DTIC


1.0
4.5
5.0
5.6
6.3
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

```
/*
   Process the user selection from the menu list.
*/
do
   {
   if (ch == 'q' || ch == 'Q' )
   {
      clrscr() .
      exit(0) ;
   }

   /*
      Initialize the graphics system.
   */
   detectgraph ( &driver , &mode) ;
   initgraph ( &driver, &mode ,NULL) ;
   xmax = getmaxx() ;
   ymax = getmaxy() ;

   /*
      Size the variables based on the number of pixels.
   */
   radius = ymax * 0.238 ;
   header = ymax * 0.15 ;
   footer = ymax * 0.95 ;
   gap = ymax * 0.025 ;
   height = ymax * 0.275 ;

   /*
      Sets the aspect ratio so tha the circles look like circles and
      not elipses.
   */
   ratio = (float)ymax/(float)xmax * 10000 * 4 /3 ;
   setaspectratio((int) ratio, 10000);
   ratio = 10000/ratio ;
   c_radius = radius * ratio ;

   /*
      Draws the outline of the venn diagram.
   */
   draw() ;
```

```
/*
   Displays the title that corresponds to the user menu selection.
*/
title() ;

/*
   Displays the message in the lower left hand corner.
*/
moveto(gap,footer+gap) ;
outtext(" Q = quit") ;

/*
   Fills the regions that correspond to the user menu selction.
*/
switch(ch)
{
case ’A’ :/* A’ intersect B’ intersect C’ */
case ’a’ :

   floodfill(xposit8,yposit8,WHITE) ;
   break;
case ’B’ :/* A intersect B intersect C */
case ’b’ :
   floodfill(xposit7,yposit7,WHITE) ;
   break;
case ’C’ :/* A’ intersect B intersect C */
case ’c’ :
   floodfill(xposit6,yposit6,WHITE) ;
   break;
case ’D’ :/* A intersect B intersect C’*/
case ’d’ :
   floodfill(xposit5,yposit5,WHITE) ;
   break;
case ’E’ :/* A intersect B’ intersect C */
case ’e’ :
   floodfill(xposit4,yposit4,WHITE) ;
   break;
case ’F’ :/* C intersect (A union B)’ */
case ’f’ :
   floodfill(xposit3,yposit3,WHITE) ;
   break;
```

```
case 'G' :/* B intersect (C union A)'*/
case 'g' :
   floodfill(xposit2,yposit2,WHITE) ;
   break;
case 'H' :/* A intersect (B union C)' */
case 'h' :
   floodfill(xposit1,yposit1,WHITE) ;
   break;
case 'I' :/* A union B union C */
case 'i' :
   floodfill(xposit1,yposit1,WHITE) ;
   floodfill(xposit2,yposit2,WHITE) ;
   floodfill(xposit3,yposit3,WHITE) ;
   floodfill(xposit4,yposit4,WHITE) ;
   floodfill(xposit5,yposit5,WHITE) ;
   floodfill(xposit6,yposit6,WHITE) ;
   floodfill(xposit7,yposit7,WHITE) ;
   break;
case 'J' :/* A' intersect B' */
case 'j' :
   floodfill(xposit3,yposit3,WHITE) ;
   floodfill(xposit8,yposit8,WHITE) ;
   break;
case 'K' :/* B' intersect C' */
case 'k' :
   floodfill(xposit1,yposit1,WHITE) ;
   floodfill(xposit8,yposit8,WHITE) ;
   break;
case 'L' :/* A  union B' */
case 'l' :
   floodfill(xposit1,yposit1,WHITE) ;
   floodfill(xposit3,yposit3,WHITE) ;
   floodfill(xposit4,yposit4,WHITE) ;
   floodfill(xposit5,yposit5,WHITE) ;
   floodfill(xposit7,yposit7,WHITE) ;
   floodfill(xposit8,yposit8,WHITE) ;
   break;
```

```
case 'M' :/* B  union C' */
case 'm' :
   floodfill(xposit1,yposit1,WHITE) ;
   floodfill(xposit2,yposit2,WHITE) ;
   floodfill(xposit5,yposit5,WHITE) ;
   floodfill(xposit6,yposit6,WHITE) ;
   floodfill(xposit7,yposit7,WHITE) ;
   floodfill(xposit8,yposit8,WHITE) ;
   break;
case 'N' :/* A  union C' */
case 'n' :
   floodfill(xposit1,yposit1,WHITE) ;
   floodfill(xposit2,yposit2,WHITE) ;
   floodfill(xposit4,yposit4,WHITE) ;
   floodfill(xposit5,yposit5,WHITE) ;
   floodfill(xposit7,yposit7,WHITE) ;
   floodfill(xposit8,yposit8,WHITE) ;
   break;
case 'O' :/* B  union C  */
case 'o' :
   floodfill(xposit2,yposit2,WHITE) ;
   floodfill(xposit3,yposit3,WHITE) ;
   floodfill(xposit4,yposit4,WHITE) ;
   floodfill(xposit5,yposit5,WHITE) ;
   floodfill(xposit6,yposit6,WHITE) ;
   floodfill(xposit7,yposit7,WHITE) ;
   break;
}/*END SWITCH*/

/*
   Process sthe user selection to quit or continue.
*/
ch = getch() ;
closegraph() ;
if (ch == 'q' || ch == 'Q')
{
   clrscr() :
   continue :
}
```

```
        /*
           Print to screen menu list
        */
        puttext(0,0,80,25,textbuff) ;
        ch = getch() ;

     }/* END do while */
     while(ch != 'q' && ch != 'Q') ;
     clrscr() ;

}/* END MAIN */
```

```
/***********************************************************************

 FUNCTION :        draw
 CALLED BY:        venninfo
 CALLS    :        cleardevice
                   settextstyle
                   setcolor
                   rectangle
                   line
                   moveto
                   outtext
                   setfillstyle
 MODIFIED :        4/12/90
 PERSON   :        Keith Calcote
 PURPOSE  :        Draws the venn diagram

***********************************************************************/

draw()
{
  /*
    Initializes the screen and sets the style and color defaults.
  */
  cleardevice() ;
  settextstyle(DEFAULT_FONT,HORIZ_DIR,1) ;
  setcolor(WHITE) ;

  /*
    Draws the outline for the diagram.
  */
  rectangle(left, top, xmax, ymax) ;
  line(0,header,xmax,header) ;
  line(0,footer,xmax,footer) ;

  /*
    Identifies the eight regions on the venn diagram by their
    x,y coordinates.
  */
  xposit1 = xmax /2 ;
  yposit1 = radius + header + gap ;

  xposit2 = xposit1 + radius * ratio * 2/3 ;
  yposit2 = yposit1 + height ;
```

```
    xposit3 = xposit1 - radius * ratio * 2/3 ;
    yposit3 = yposit2 ;

    xposit4 = xposit1 - radius * ratio /3 ;
    yposit4 = yposit1 + radius /sqrt(3) ;

    xposit5 = xposit1 + radius * ratio /3 ;
    yposit5 = yposit4 ;

    xposit6 = xposit1 ;
    yposit6 = yposit2 ;

    xposit7 = xposit1 ;
    yposit7 = yposit1 + radius * 2/3 ;

    xposit8 = 2 * gap * ratio ;
    yposit8 = header + 2 * gap ;

    /*
      Draws the three circles for the venn diagram.
    */
    circle(xposit1,yposit1,c_radius) ;
    circle(xposit2,yposit2,c_radius) ;
    circle(xposit3,yposit3,c_radius) ;

    /*
      Identifies the three regions on the venn diagram as A, B and C.
    */
    settextstyle(DEFAULT_FONT,HORIZ_DIR,2) ;
    moveto(xposit1 , yposit1-radius/2) ;
    outtext("A") ;
    moveto(xposit2 + radius/2 , yposit2+ 2*gap) ;
    outtext("B") ;
    moveto(xposit3 - radius/2, yposit3 + 2*gap) ;
    outtext("C") ;

    /*
      Sets the text style back to default.
    */
    settextstyle(DEFAULT_FONT,HORIZ_DIR,1) ;
    setfillstyle(LTSLASH_FILL,WHITE) ;
}
```

```
/*********************************************************************

 FUNCTION :      title
 CALLED BY:      venninfo
 CALLS   :       moveto
                 settextjustify
                 settextstyle
                 outtext
 MODIFIED :      4/12/90
 PERSON  :       Keith Calcote
 PURPOSE  :      Displays the title of the selected venn diagram.

*********************************************************************/

title()
{
  /*
    Moves the cursor position to the title area.
  */
  moveto(xmax /2, header /2) ;
  settextjustify(CENTER_TEXT,CENTER_TEXT) ;
  settextstyle(DEFAULT_FONT,HORIZ_DIR,2) ;

  /*
    The title is displyed based upon the user's menu selection.
  */
  switch(ch)
  {
  case 'A' :
  case 'a' :
    outtext("A' intersect B' intersect C'") ;
    break;

  case 'B' :
  case 'b' :
    outtext("A intersect B intersect C") ;
    break;

  case 'C' :
  case 'c' :
    outtext("A' intersect B intersect C") ;
    break;
```

```
case 'D' :
case 'd' :
  outtext("A intersect B intersect C'") ;
  break;

case 'E' :
case 'e' :
  outtext("A intersect B' intersect C") ;
  break;

case 'F' :
case 'f' :
  outtext("C intersect (B union A)'") ;
  break;

case 'G' :
case 'g' :
  outtext("B intersect (C union A)'") ;
  break;

case 'H' :
case 'h' :
  outtext("A intersect (B union C)'") ;
  break;

case 'I' :
case 'i' :
  outtext("A union B union C") ;
  break;

case 'J' :
case 'j' :
  outtext("A' intersect B'") ;
  break;



case 'K' :
case 'k' :
  outtext("B' intersect C'") ;
  break;
  case 'L' :
  case 'l' :
    outtext("A  union B'") ;
    break;


  case 'M' :
  case 'm' :
    outtext("B  union C'") ;
    break;


  case 'N' :
  case 'n' :
    outtext("A  union C'") ;
    break;


  case 'O' :
  case 'o' :
    outtext("B  union C") ;
    break;



  case 'Q' :
  case 'q' :
    clrscr() ;
    exit(0) ;
    break ;

  default :
    break ;

  }/* END switch */

  settextjustify(LEFT_TEXT,TOP_TEXT) ; /* default settings */
  settextstyle(DEFAULT_FONT,HORIZ_DIR,1) ;

}/* END title() */
```

APPENDIX O

THE CODE: FILE "RULES.C"

```
/*************************************************************************
                    The Discrete Math Tutor (DMT)
             Thesis Project at the Naval Postgraduate School
                 1989-1990 by Keith Calcote and Rick Howard

 FILENAME: rules.c

 LIBRARY CALLS:
              exit              Turbo C Lib
              waitkey           CXL Lib
              wcclear           CXL Lib
              whelpcat          CXL Lib
              whelpdef          CXL Lib
              wopen             CXL Lib
              wprintf           CXL Lib
              wshadow           CXL Lib
              wtitle            CXL Lib
              set_video         DMT Utilities

 PROGRAM CALLS:
              NONE

 RULES  FUNCTIONS:
              and
              iff
              imply
              or
              pre_help
              quit

 COMPLETED:      4/12/90

 PERSONS:        Keith Calcote & Rick Howard

 PURPOSE:        Displays a quick reference truth table for the following
                 logic expressions: AND, OR, IMPLIES and IF-&-ONLY-IF
```

```
*******************************************************************/

/* header files *`

#include <stdio.h>
#include "d:\cxl\cxlwin.h"
#include "d:\cxl\cxlkey.h"
#include "d:\cxl\cxlvid.h"
#include "d:\tc\thesis\video.h"
#include "d:\tc\thesis\help.h"
/*---------------------------------------------------------------*/

/* function prototypes */

static void and(void);
static void or(void);
static void imply(void);
static void iff(void);
static void quit(void);
static void pre_help(void);
/*---------------------------------------------------------------*/

/* Macros */

#define HEADING wprintf("    P  \xb3  Q   \xb3\xb3 ")
#define FF wprintf("    F  \xb3  F   \xb3\xb3 ")
#define FT wprintf("    F  \xb3  T   \xb3\xb3 ")
#define TF wprintf("    T  \xb3  F   \xb3\xb3 ")
#define TT wprintf("    T  \xb3  T   \xb3\xb3 ")
#define LINE wprintf("\x20\x20\x20\xc4\xc4\xc4\xc4\xc4\xc4\xc5\xc4")
#define LINEP wprintf("\xc4\xc4\xc4\xc4\xc4\xc4\xc4\xc5\xc5\xc4\xc4\xc4")
#define LINEPP wprintf("\xc4\xc4\xc4\xc4\xc4\xc4\xc4\xc4\xc4\n")
#define TRUE wprintf("     T\n")
#define FALSE wprintf("     F\n")
#define CLR wcclear(WHITE|_CYAN);
#define SPACE wprintf("\n");
/*---------------------------------------------------------------*/
```

```
/*******************************************************************

 FUNCTION :       main
 CALLED BY:       NONE
 CALLS   :        See Declarations
 MODIFIED :       4/12/90
 PERSON  :        Rick Howard & Keith Calcote
 PURPOSE :        See Declarations

*******************************************************************/

main()
{
  /*
    Check for mono, CGA or EGA screen.
  */
  set_video();

  /*
    Define all hot-keys.
  */
  setonkey(0x3B00,and,0);           /* F1 */
  setonkey(0x3C00,or,0);            /* F2 */
  setonkey(0x3D00,imply,0);         /* F3 */
  setonkey(0x3E00,iff,0);           /* F4 */
  setonkey(0x011B,quit,0);          /* ESC  */

  /*
    Open a window for the truth table.
  */
  if(!wopen(2,42,11,77,3,WHITE|_CYAN,WHITE|_CYAN)) quit();
  wtitle("[ F1-AND  F2-OR  F3-IMPLY  F4-IFF ]",TCENTER,BLUE|_CYAN);
  wshadow(LGREY|_BLACK);

  /*
    Define the help screen attributes.
  */

whelpdef("DMT.HLP",0x2368,BLACK|_LGREY,BLACK|_LGREY,LBLUE|_LGREY,
                    LRED|_LGREY,pre_help);
    /*
      Set the help screen that applies to any generic lesson.
    */
    whelpcat(H_TRUTH_TABLE_RULES);

    /*
      Wait for the user's response.
    */
    while (waitkey() != 0x4C35);
    wprintf("Error = %s\n",werrmsg());
}
```

```
/*********************************************************************

 FUNCTION :      and
 CALLED BY:      rules
 CALLS    :      wprintf
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard & Keith Calcote
 PURPOSE  :      Displays the truth table for the logic expression "AND"

*********************************************************************/

static void and(void)
{
  CLR;
  SPACE;
  HEADING ;
  wprintf("P AND Q\n") ;
  LINE;
  LINEP;
  LINEPP;
  TT ;
  TRUE;
  TF ;
  FALSE;
  FT ;
  FALSE;
  FF ;
  FALSE;
}
```

```
/*********************************************************************

 FUNCTION :      or
 CALLED BY:      rules
 CALLS    :      wprintf
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard & Keith Calcote
 PURPOSE  :      Displays the truth table for the logic expression "OR"

*********************************************************************/

static void or(void)
{
  CLR;
  SPACE;
  HEADING ;
  wprintf("P OR Q\n") ;
  LINE;
  LINEP;
  LINEPP;
  TT ;
  TRUE;
  TF ;
  TRUE;
  FT ;
  TRUE;
  FF ;
  FALSE;
}
```

```
/********************************************************************

 FUNCTION :      imply
 CALLED BY:      rules
 CALLS    :      wprintf
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard & Keith Calcote
 PURPOSE  :      Displays the truth table for the logic expression "IMPLY"

********************************************************************/

static void imply(void)
{
  CLR;
  SPACE;
  HEADING ;
  wprintf("P IMPLIES Q\n") ;
  LINE;
  LINEP;
  LINEPP;
  TT ;
  TRUE;
  TF ;
  FALSE;
  FT ;
  TRUE;
  FF ;
  TRUE;
}
```

```
/**********************************************************************

 FUNCTION :       iff
 CALLED BY:       rules
 CALLS    :       wprintf
 MODIFIED :       4/12/90
 PERSON   :       Rick Howard & Keith Calcote
 PURPOSE  :       Displays the truth table for the logic expression "IFF"

**********************************************************************/

static void iff(void)
{
  CLR;
  SPACE;
  HEADING ;
  wprintf("P IFF Q\n") ;
  LINE;
  LINEP;
  LINEPP;
  TT ;
  TRUE;
  TF ;
  FALSE;
  FT ;
  FALSE;
  FF ;
  TRUE;
}
```

```
/**********************************************************************

 FUNCTION :      quit
 CALLED BY:      rules
 CALLS    :      wprintf
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard & Keith Calcote
 PURPOSE  :      Terminate the program

**********************************************************************/

static void quit(void)
{
  exit(0);
}

/**********************************************************************

 FUNCTION :      pre_help
 CALLED BY:      rules
 CALLS    :      wshadow
                 setonkey
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard & Keith Calcote
 PURPOSE  :      draws a shadow behind the open window

**********************************************************************/

static void pre_help(void)
{
  wshadow(LGREY|_BLACK);
}
```

APPENDIX P

THE CODE: FILE "TABLE.C"

```
/***********************************************************************
                 The Discrete Math Tutor (DMT)
          Thesis Project at the Naval Postgraduate School
           1989-1990 by Keith Calcote and Rick Howard

LIBRARY CALLS:
               atoi            Turbo C Lib
               exit            Turbo C Lib
               ltoa            Turbo C Lib
               setonkey        CXL Lib
               set_video       DMT Utilities
               strcat          Turbo C Lib
               strcmp          Turbo C Lib
               strcpy          Turbo C Lib
               strlen          Turbo C Lib
               strupr          Turbo C Lib
               waitkey         CXL Lib
               wcclear         CXL Lib
               wcenters        CXL Lib
               wgets           CXL Lib
               wgotoxy         CXL Lib
               whelpcat        CXL Lib
               whelpdef        CXL Lib
               wopen           CXL Lib
               wprintf         CXL Lib
               wshadow         CXL Lib

PROGRAM CALLS:
               NONE
```

```
TABLE FUNCTIONS:
            and
            display_loop
            error_response
            findllhcol
            findrhcol
            iff
            imply
            negation
            or
            pause
            pre_help
            quit
            replstr
            update_name

COMPLETED:      4/12/90

PERSONS:        Keith Calcote & Rick Howard

PURPOSE:        Displays the truth table to any user supplied logic
                equation

**********************************************************************/

/* header files */

#include <stdio.h>
#include <process.h>
#include <stdlib.h>
#include <string.h>
#include "d:\cxl\cxlwin.h"
#include "d:\cxl\cxlkey.h"
#include "d:\cxl\cxlvid.h"
#include "d:\tc\thesis\video.h"
#include "d:\tc\thesis\help.h"
/*------------------------------------------------------------------*/
```

```
/* Constants */

#define LEN 81
#define NUMLABEL 81
#define NUMELE 20
#define NUMCOL 40
#define CLR wcclear(WHITE|_CYAN);
#define INPUT_ERROR 1
#define PAREN_MISMATCH 2
#define INVALID_CHAR 3
/*----------------------------------------------------------------*/

/* Type Definitions */

struct table
{
   char element[NUMELE] ;
   char label[NUMLABEL] ;
   char name[LEN] ;
};

struct table col[NUMCOL] ;
/*----------------------------------------------------------------*/

/* globals */

char nextcol[NUMLABEL],lastcol[NUMLABEL] ;
char numstrng[LEN] ;
int length,numcols,lhcol,rhcol ;
long lvalue ;
/*----------------------------------------------------------------*/
```

```
/* function prototypes */

static char and(char[LEN]) ;
static char or(char[LEN]) ;
static char imply(char[LEN]) ;
static char iff(char[LEN]) ;
static void error_response(int num);
static void pause(void);
static void quit(void);
static void display_loop(void);
static void pre_help(void);
static void updatename(char str[], char origstr[]);
static void negations(char str[]);
static void findlhdcol(char str[], int addr);
static void findrhcol(char str[], int addr);
static void replstr(char str[], char loc[], chr rep[]);
/*----------------------------------------------------------------------------*/
```

```
/*******************************************************************

 FUNCTION :      main
 CALLED BY:      NONE
 CALLS    :      See Declarations
 MODIFIED :      4/12/90
 PERSON   :      Rick Howard & Keith Calcote
 PURPOSE  :      See Declarations

*******************************************************************/

main()
{
  WINDOW w;       /* Window handle */

  /*
     Define the hot-key for this program.
  */
  setonkey(0x011B,quit,0);

  /*
     Check for mono, CGA or EGA screen.
  */
  set_video();

  /*
     Open a window to display the program.
  */
  if((w=wopen(2,0,23,79,3,WHITE|_CYAN,WHITE|_CYAN))==0)
    wprintf("error_exit(1);");

  /*
     Define the help screen attributes.
  */
  whelpdef("DMT.HLP",0x2368,BLACK|_LGREY,BLACK|_LGREY,
                                    LBLUE|_LGREY,LRED|_LGREY,pre_help);

  /*
     Set the truth table help screen in place.
  */
  whelpcat(H_TRUTH_TABLE_PROBLEM_SOLVER);
  /*
      Present the title screen to the program.
  */
  CLR;
  wcenters(1,YELLOW|_BROWN,"Welcome to the truth table generator" ) ;

  /*
      Begin the main program.
  */
  while(1)
     display_loop();
}
```

```
/*********************************************************************

 FUNCTION :      display_loop
 CALLED BY:      table
 CALLS   :       strcpy
                 wcenters
                 wgotoxy
                 wgets
                 strupr
                 strcpy
                 ltoa
                 strcmp
                 replstr
                 negation
                 and
                 or
                 imply
                 iff
                 strcat
                 wprintf
 MODIFIED :      4/12/90
 PERSON  :       Rick Howard & Keith Calcote
 PURPOSE  :      The main loop of the table, allows the user to cycle through
                 as many truth tables as he desires

*********************************************************************/

static void display_loop(void)
{
  char key ;                /* Holds the user's input              */

  char input[LEN];          /* User's desired equation              */

  char temp1[LEN];          /* Temorary string storage              */
  char temp2[LEN];
  char temp3[LEN];

  char nulline[LEN] ;       /* NULL String                          */

  char replacenum[10] ;     /* Temporary string storage             */

  int dex,n ;               /* Counters                             */
int openpar = 0 ;          /* Total number of open parens            */

int closepar = 0 ;         /* Total number of closed parens          */

int pflag = 0 ;            /* Set if "p" is used as a proposition    */

int qflag = 0 ;            /* Set if "q" is used as a proposition    */

int rflag = 0 ;            /* Set if "r" is used as a proposition    */

int sflag = 0 ;            /* Set if "s" is used as a proposition    */

int frstclspar ;           /* The location of the first closed paren */

int frstopenpar ;          /* The location of the first open paren   */

int sumflag ;              /* Total number of different propositions used */

int breakflag ;            /* Flag used to break out of a while loop */

/*
   Wait for the user to press any key.
*/
pause();
CLR;

/*
   Nullifies the struct table col[].
*/
for(dex = 0; dex < NUMCOL; dex ++)
{
  for(n = 0; n < NUMELE; n ++)
    col[dex].element[n] = '\x00' ;
  for(n = 0; n < NUMLABEL; n ++)
    col[dex].label[n] = '\x00' ;
  for(n = 0; n < LEN; n ++)
    col[dex].name[n] = '\x00' ;
}
```

```
/*
   Sets the null string to NULL.
*/
for(dex = 0; dex < LEN; dex++)
   nulline[dex] = '\x00' ;

/*
   Initialiizes all strings to NULL.
*/
strcpy(input,nulline) ;
strcpy(temp1,nulline) ;
strcpy(temp2,nulline) ;
strcpy(temp3,nulline) ;
strcpy(replacenum,nulline) ;

/*
   Get the user's logic equation and converts to upper case.
*/
wcenters(0,WHITE|_CYAN,"Enter the equation for the truth table:");
wgotoxy(2,0);
wgets(input) ;
strupr(input) ;

/*
   Removes blanks from the input equation.
*/
for (dex = 0; dex < strlen(input); dex++)
{
   if(input[dex] == ' ')
   {
      strcpy(&input[dex], &input[dex+1]) ;
      dex -- ;
   }
}
```

```
/*
   Examines each element of the input string for possible errors.
*/
for (dex=0; dex < strlen(input); dex++)
{
  switch(input[dex])
  {

  case '(' :
     switch(input[dex+1])
     {
     case ')' :
     case '&' :
     case '|' :
     case '>' :
     case '=' :
        error_response(INPUT_ERROR);
     default :
        break ;
     }
     openpar++ ;
     break ;

  case ')' :
     switch(input[dex+1])
     {
     case '(' :
     case 'R' :
     case 'P' :
     case 'Q' :
     case 'S' :
        error_response(INPUT_ERROR);
     default :
        break ;
     }
     closepar++ ;
     break ;
```

```
case '~' :
   switch(input[dex+1])
   {
   case ')' :
   case '&' :
   case '|' :
   case '>' :
   case '=' :
      error_response(INPUT_ERROR);
   default :
      break ;
   }
   break ;

case 'P' :
   switch(input[dex+1])
   {
   case '(' :
   case '~' :
   case 'R' :
   case 'P' :
   case 'Q' :
   case 'S' :
      error_response(INPUT_ERROR);
   default :
      break ;
   }
   pflag = 1 ;
   break ;
```

```
case 'Q' :
   switch(input[dex+1])
   {
   case '(' :
   case '~' :
   case 'R' :
   case 'P' :
   case 'Q' :
   case 'S' :
      error_response(INPUT_ERROR);
   default :
      break ;
   }
   qflag = 1 ;
   break ;

case 'R' :
   switch(input[dex+1])
   {
   case '(' :
   case '~' :
   case 'R' :
   case 'P' :
   case 'Q' :
   case 'S' :
      error_response(INPUT_ERROR);
   default :
      break ;
   }
   rflag = 1 ;
   break ;
```

```
    case 'S' :
        switch(input[dex+1])
        {
        case '(' :
        case '~' :
        case 'R' :
        case 'P' :
        case 'Q' :
        case 'S' :
            error_response(INPUT_ERROR);
        default :
            break ;
        }
        sflag = 1 ;
        break ;

    case '&' :
    case '|' :
    case '>' :
    case '=' :
        switch(input[dex+1])
        {
        case ')' :
        case '&' :
        case '|' :
        case '>' :
        case '=' :
        case '\x00' :
            error_response(INPUT_ERROR);
        default :
            break ;
        }
        break ;
    case ' ' :
        break ;

    default:
        error_response(INVALID_CHAR);
    }
}
```

```
/*
   Check for paren errors.
*/
if(openpar != closepar)
{
  error_response(PAREN_MISMATCH);
}

/*
   Determines how many  of the possible variables are used.
*/
sumflag = pflag + qflag + rflag + sflag ;
if(sumflag == 0 )
{
  quit();
}
numcols = sumflag ;

/*
   Create initial columns.
*/
switch(sumflag)
{
  case (4) :
  length = 16 ;
  strcpy(col[0].label,"P") ;
  strcpy(col[1].label,"Q") ;
  strcpy(col[2].label,"R") ;
  strcpy(col[3].label,"S") ;
  strcpy(col[0].element,"TTTTTTTTFFFFFFFF") ;
  strcpy(col[1].element,"TTTTFFFFTTTTFFFF") ;
  strcpy(col[2].element,"TTFFTTFFTTFFTTFF") ;
  strcpy(col[3].element,"TFTFTFTFTFTFTFTF") ;
  break ;
  case (3) :
  length = 8 ;
  if(pflag == 0)
  {
    strcpy(col[0].label,"Q") ;
    strcpy(col[1].label,"R") ;
    strcpy(col[2].label,"S") ;
  }
```

```
if(qflag == 0)
{
   strcpy(col[0].label,"P") ;
   strcpy(col[1].label,"R") ;
   strcpy(col[2].label,"S") ;
}
if(rflag == 0)
{
   strcpy(col[0].label,"P") ;
   strcpy(col[1].label,"Q") ;
   strcpy(col[2].label,"S") ;
}
if(sflag == 0)
{
   strcpy(col[0].label,"P") ;
   strcpy(col[1].label,"Q") ;
   strcpy(col[2].label,"R") ;
}
strcpy(col[0].element,"TTTTFFFF") ;
strcpy(col[1].element,"TTFFTTFF") ;
strcpy(col[2].element,"TFTFTFTF") ;
break ;
case(2) :
length = 4 ;
if(pflag == 0 && qflag == 0)
{
   strcpy(col[0].label,"R") ;
   strcpy(col[1].label,"S") ;
}
if(pflag == 0 && rflag == 0)
{
   strcpy(col[0].label,"Q") ;
   strcpy(col[1].label,"S") ;
}
if(pflag == 0 && sflag == 0)
{
   strcpy(col[0].label,"Q") ;
   strcpy(col[1].label,"R") ;
}
```

```
   if(qflag == 0 && rflag == 0)
   {
      strcpy(col[0].label,"P") ;
      strcpy(col[1].label,"S") ;
   }
   if(qflag == 0 && sflag == 0)
   {
      strcpy(col[0].label,"P") ;
      strcpy(col[1].label,"R") ;
   }
   if(rflag == 0 && sflag == 0)
   {
      strcpy(col[0].label,"P") ;
      strcpy(col[1].label,"Q") ;
   }
   strcpy(col[0].element,"TTFF") ;
   strcpy(col[1].element,"TFTF") ;
   break ;
   case(1) :
   length = 2 ;
   if(pflag == 1)
      strcpy(col[0].label,"P") ;
   if(qflag == 1)
      strcpy(col[0].label,"Q") ;
   if(rflag == 1)
      strcpy(col[0].label,"R") ;
   if(sflag == 1)
      strcpy(col[0].label,"S") ;
   strcpy(col[0].element,"TF") ;
   break ;
}

/*
   Copies all labels into the names of the structures col[].
*/
for(dex = 0; dex < numcols; dex ++ )
   strcpy(col[dex].name,col[dex].label) ;
```

```
/*
   Create numstrng with col numbers.
*/
strcpy(numstrng,input) ;
for (lvalue = 0; lvalue < numcols; lvalue ++ )
{
   ltoa(lvalue,replacenum,10) ;
   if(strcmp(col[lvalue].label,"P") == 0)
      replstr(numstrng,col[lvalue].label,replacenum) ;
   if(strcmp(col[lvalue].label,"Q") == 0)
      replstr(numstrng,col[lvalue].label,replacenum) ;
   if(strcmp(col[lvalue].label,"R") == 0)
      replstr(numstrng,col[lvalue].label,replacenum) ;
   if(strcmp(col[lvalue].label,"S") == 0)
      replstr(numstrng,col[lvalue].label,replacenum) ;
}

/*
   Checks the input string for negations.
*/
negation(numstrng) ;

/*
   Operates on the expressions inside parens.
*/
breakflag = 0 ;
for( n=0; n < strlen(numstrng); n++)
{
   dex = 0 ;

   /*
      Counts the number of elements up to the first closed paren.
   */
   while(numstrng[dex] != ')' )
   {
      dex ++ ;
      if(dex > strlen(numstrng) )
      {
         breakflag = 1 ;
         break ;
      }
   }
```

```
if(breakflag == 1)
   break ;
frstclspar = dex + 1 ;

/*
    Find matching paren.
*/
while(numstrng[dex] != '(' && dex >= 0 )
   dex -- ;
frstopenpar = dex ;

/*
    Checks the location of the closed paren.  If at the end of the
    input string, then NULL is assigned to temp1.  Otherwise, temp1
    is assigned the string to the right of the closed paren.
*/
if(frstclspar + 1 > strlen(numstrng) )
   strcpy ( temp1,nulline ) ;
else
   strcpy ( temp1, &numstrng[frstclspar] ) ;

/*
   Breaks up the input string so that strings inside parens may be
   isolated.
*/
strcpy ( temp2, numstrng ) ;
strcpy ( &temp2[frstopenpar], nulline ) ;
strcpy ( temp3, numstrng ) ;
strcpy ( &temp3[frstclspar - 1], nulline ) ;
strcpy ( temp3, &temp3[frstopenpar + 1] ) ;

/*
    Associated functions operate on the string inside the paren.
*/
and(temp3) ;
or(temp3) ;
imply(temp3) ;
iff(temp3) ;
```

```
    /*
       Copies the relationship iside the paren to the col[].name
       structure element.  Then, reduces the value in temp1 to a
       number corresponding to the next column.
    */
    strcpy(temp1,"(" ) ;
    strcat(temp1,&col[numcols - 1].name) ;
    strcat(temp1,")" ) ;
    strcpy(&col[numcols-1].name,temp1) ;
    strcpy(temp1,"(" ) ;
    strcat(temp1,nextcol) ;
    strcat(temp1,")" ) ;

    /*
       Removes the paren from numstring.
    */
    replstr(numstrng,temp1,nextcol) ;
}

/*
   Performs the associated functions on numstring.
*/
negation(numstrng) ;
and(numstrng) ;
or(numstrng) ;
imply(numstrng) ;
iff(numstrng) ;

/*
   Displays the propositions on the screen.
*/
dex = 1 ;
for(n = sumflag; n < numcols; n ++)
{
    strcpy(col[n].label,"P") ;
    lvalue = dex ;
    ltoa(lvalue,temp1,10) ;
    strcat(col[n].label,temp1) ;
    wprintf("%s : %s\n",col[n].label, col[n].name) ;
    dex ++ ;
}
wprintf("\n") ;
```

```
  /*
     Displays the truth table column heading to the screen.
  */
  for(n = 0; n < numcols; n ++)
     if(strlen(col[n].label) > 2)
        wprintf("%-4s",col[n].label) ;
  else
     wprintf("%-3s",col[n].label) ;
  wprintf("\n") ;

  /*
     Displays the truth table.
  */
  for(n = 0; n < length; n ++)
  {
     for(dex = 0; dex < numcols; dex ++)
     {
        strcpy(temp1,&col[dex].element[n]) ;
        strcpy(&temp1[1],"\x00") ;
        if(strlen(col[dex].label) > 2)
           wprintf(" %-3s",temp1) ;
        else
           wprintf("%-3s",temp1) ;
     }
     wprintf("\n") ;
  }

}
```

```
/*******************************************************************************

 FUNCTION :      replstr
 CALLED BY:      display_loop
                 negation
                 updatename
 CALLS   :       strcpy
                 strcat
 MODIFIED :      4/12/90
 PERSON  :       Keith Calcote
 PURPOSE :       Replaces alphabetic characters with numeric characters

*******************************************************************************/

static void replstr(char str[],chr loc[],chr rep[])
{
  char temp1[81] ;    /*  Temporary string storage              */

  int dex,ind,test ;  /*  Counters                      */

  /*
     Replaces all occurances of loc[] in str[] with rep[].
  */
  for ( dex = 0; dex <= ( strlen(str)-strlen(loc) ); dex ++ )
  {
    /*
       Prevents the loop from exiting under listed conditions:
       needed because the length of str may change the operation
       the loop.
    */
    if(dex < 0 || dex > strlen(str) ) break ;

    /*
        Locates and replaces the desired string.
    */
    if( (char)str[dex] == (char)loc[0] )
      {
      test = 0 ;
```

```
        /*
           Increments test for each matching character.
        */
        for(ind = 0; ind < strlen(loc); ind ++ )
        {
           if( (char)str[ind + dex] == (char)loc[ind] )
              test ++ ;
        }

        /*
             If the entire string is matched, then it is replaced.
        */
        if( test == strlen(loc) )
           {
           strcpy(temp1,&str[dex+test]) ;
           strcpy(&str[dex],rep) ;
           strcat(str,temp1) ;
           }
      }
   }
}
```

```
/*********************************************************************

 FUNCTION :       findrhcol
 CALLED BY:       negation
                  and
                  or
                  imply
                  iff
 CALLS   :        strcpy
                  atoi
 MODIFIED :       4/12/90
 PERSON  :        Keith Calcote
 PURPOSE  :       Identifies and returns the numeric value of the col which is
                  on the imeediate right hand side of the operator.

*********************************************************************/

findrhcol(char str[], int addr)
{
  int rhc ;                  /* Numeric value of the col on the right
                                hand side of the operator          */

  char findtemp [LEN] ;      /* Temporary character storage         */


  strcpy(findtemp,&str[addr]) ;
  rhc = atoi(&str[addr+1]) ;
  return(rhc) ;
}
```

```
/*********************************************************************

  FUNCTION :        findlhcol
  CALLED BY:        negation
                    and
                    or
                    imply
                    iff
  CALLS    :        strcpy
                    atoi
  MODIFIED :        4/12/90
  PERSON   :        Keith Calcote
  PURPOSE  :        Finds the left hand col numerical value.

*********************************************************************/

findlhcol(char str[], int addr)
{
  int lhc ;                 /*  Numerical value of the col to the left
                                the operator                 */

  char findtemp [LEN] ;     /*  Temporary string storage          */

  /*
     Places the incoming string into temporary storage and places a
     NULL character at the end.
  */
  strcpy(findtemp,str) ;
  findtemp[addr+1] = '\x00' ;

  /*
     Locates left hand values that are zero thru nine.
  */
  if ( strlen(findtemp) < 3 ||
     (char)str[addr-2] == '&' ||
     (char)str[addr-2] == '|' ||
     (char'str[addr-2] == '>' ||
     (char)str[addr-2] == '=' ||
     (char)str[addr-2] == '(' ||
     (char)str[addr-2] == ')' )
     {
     lhc = atoi(&str[addr-1]) ;
  }
```

```
    /*
       Locates left hand values that are greater than nine.
    */
    else
    {
      lhc = atoi( &(char)str[addr-2] ) ;
    }

    return(lhc) ;
}
```

```
/**********************************************************************

 FUNCTION :      negation
 CALLED BY:      display_loop
                 replstr
 CALLS   :       findrhcol
                 strcat
                 strcpy
                 ltoa
                 replstr
 MODIFIED :      4/12/90
 PERSON  :       Keith Calcote
 PURPOSE  :      Finds all negations on col's and updates structure with neg
                 col's

**********************************************************************/

negation(char str[])
{
  int dex, n;  /*  Counters   */

  /*
     Investigates every character in str for negations.
  */
  for ( dex = 0; dex < strlen(str); dex ++ )
  {
    /*
       Finds the column number to the immediate right of the
       negation.
    */
    if ( (char)str[dex] == '~' &&
       (char)str[dex+1] != '~' &&
       (char)str[dex+1] != '(' )
       {
       rhcol = findrhcol(str,dex) ;
```

```
      /*
         For each element in the column to the right of the
         negation, change the value from 'T' to 'F' or from
         'F' to 'T'.
      */
      for(n = 0;n <length; n ++)
         if( col[rhcol].element[n] == 'T' )
         {
         col[numcols].element[n] = 'F' ;
      }
      else
         {
         col[numcols].element[n] = 'T' ;
      }

      /*
         Create the proper name for the column and update str.
      */
      col[numcols].name[0] = '~' ;
      strcat(col[numcols].name,col[rhcol].name ) ;
      strcpy(lastcol,"~") ;
      lvalue = rhcol ;
      ltoa(lvalue,nextcol,10) ;
      strcat(lastcol,nextcol) ;
      lvalue = numcols ;
      ltoa(lvalue,nextcol,10) ;
      replstr(str,lastcol,nextcol) ;

      numcols ++ ;
    }
  }
}
```

```
/***********************************************************************

 FUNCTION :      and
 CALLED BY:      display_loop
 CALLS   :       findrhcol
                 findlhcol
                 updatename
 MODIFIED :      4/12/90
 PERSON  :       Keith Calcote
 PURPOSE :       Given string with col numbers produces AND col

***********************************************************************/

static char and(char str[])
{
  int dex, n ;              /*  Counters                    */

  char oper[LEN] = "&" ;    /*  The AND operator              */

  /*
    Investigates each character for the AND operator.
  */
  for (dex = 0; dex < strlen(str); dex ++)
  {
    /*
      Locates the column number immediately to the left and right
      of the AND operator.
    */
    if ( (char)str[dex] == '&')
      {
      rhcol = findrhcol(str,dex) ;
      lhcol = findlhcol(str,dex) ;

      /*
        Creates a new column with  values equal to the lhcol
        AND rhcol.
      */
      for (n = 0:n < length; n ++)
      {
            /*
               Determines the elements in the new row. (T or F)
            */
            if( (col[lhcol].element[n] == 'T') &&
               (col[rhcol].element[n] == 'T') )
               col[numcols].element[n] = 'T' ;
            else
               col[numcols].element[n] = 'F' ;
         }

         /*
            Updates str with new column name.
         */
         updatename(oper,str) ;
         dex -- ;
      }
   }
}
```

```
/*********************************************************************

 FUNCTION :     or
 CALLED BY:     display_loop
 CALLS   :      findrhcol
                findlhcol
                updatename
 MODIFIED :     4/12/90
 PERSON  :      Keith Calcote
 PURPOSE :      Given string with col numbers produces OR col

*********************************************************************/

static char or(char str[]);
{
  int dex, n ;                /*  Counters                    */

  char oper[LEN] = "|" ;      /*  The OR operator                 */

  /*
    Investigates each character for the OR operator.
  */
  for (dex = 0; dex < strlen(str); dex ++)
  {
    /*
      Locates the column number immediately to the left and right
      of the OR operator.
    */
    if ( (char)str[dex] == '|')
      {
      rhcol = findrhcol(str,dex) ;
      lhcol = findlhcol(str,dex) ;
```

```
          /*
             Creates a new column with  values equal to the lhcol
             OR rhcol.
          */
          for (n = 0;n < length; n ++)
          {
             if( (col[lhcol].element[n] == 'T') ||
                (col[rhcol].element[n] == 'T') )
                col[numcols].element[n] = 'T' ;
             else
                col[numcols].element[n] = 'F' ;
          }

          /*
              Updates str with new column name.
          */
          updatename(oper,str) ;
          dex -- ;
       }
    }
}
```

```
/**********************************************************************

 FUNCTION :      imply
 CALLED BY:      display_loop
 CALLS    :      findrhcol
                 findlhcol
                 updatename
 MODIFIED :      4/12/90
 PERSON   :      Keith Calcote
 PURPOSE  :      Given string with col numbers produces IMPLY col

**********************************************************************/

static char imply(char str[])
{
  int dex, n ;               /*  Counters                  */

  char oper[LEN] = ">" ;     /*  The IMPLY operator              */

  /*
    Investigates each character for the IMPLY operator.
  */
  for (dex = 0; dex < strlen(str); dex ++)
  {
    /*
      Locates the column number immediately to the left and right
      of the IMPLY operator.
    */
    if ( (char)str[dex] == '>')
      {
      rhcol = findrhcol(str,dex) ;
      lhcol = findlhcol(str,dex) ;
```

```
         /*
            Creates a new column with  values equal to the lhcol
            IMPLY rhcol.
         */
         for (n = 0;n < length; n ++)
         {
            if( (col[lhcol].element[n] == ’T’) )
               {
               if( col[rhcol].element[n] == ’T’)
                  col[numcols].element[n] = ’T’ ;
               else
                  col[numcols].element[n] = ’F’ ;
            }
            else
               col[numcols].element[n] = ’T’ ;
         }

         /*
            Updates str with new column name.
         */
         updatename(oper,str) ;
         dex -- ;
      }
   }
}
```

```
/*********************************************************************

 FUNCTION :       iff
 CALLED BY:       display_loop
 CALLS    :       findrhcol
                  findlhcol
                  updatename
 MODIFIED :       4/12/90
 PERSON   :       Keith Calcote
 PURPOSE  :       Given string with col numbers produces iff col

*********************************************************************/

static char iff(char str[])
{
  int dex, n ;                        /*  Counters                   */

  char oper[LEN] = "=" ;              /*  The IFF operator             */

  char ltemp[LEN],rtemp[LEN] ;        /*  Temporary character storage for
                                           left hand and right hand side of
                                           the operator              */

  /*
    Investigates each character for the IFF operator.
  */
  for (dex = 0; dex < strlen(str); dex ++)
  {
    /*
      Locates the column number immediately to the left and right
      of the IFF operator.
    */
    if ( (char)str[dex] == '=')
      {
      rhcol = findrhcol(str,dex) ;
      lhcol = findlhcol(str,dex) ;
```

```
        /*
           Creates a new column with  values equal to the lhcol
           IFF rhcol.
        */
        for (n = 0;n < length; n ++)
        {
           /*
              Isolates the individual row elements.
           */
           strcpy(ltemp,&col[lhcol].element[n] ) ;
           ltemp[1] = '\x00' ;
           strcpy(rtemp,&col[rhcol].element[n] ) ;
           rtemp[1] = '\x00' ;

           /*
               Performs the IFF operation
           */
           if( strcmp(ltemp;rtemp) == 0 )
              col[numcols].element[n] = 'T' ;
           else
              col[numcols].element[n] = 'F' ;
        }
        /*
           Updates str with new column name.
        */
        updatename(oper,str) ;
        dex -- ;
     }
  }
}
```

```
/*********************************************************************

 FUNCTION :     updatename
 CALLED BY:     display_loop
                imply
                iff
                and
                or
 CALLS   :      strcpy
                strcat
                ltoa
 MODIFIED :     4/12/90
 PERSON  :      Keith Calcote
 PURPOSE  :     Update col[] name give the operator as input string
                also updates numstmg with number of new col

*********************************************************************/

static void updatename(char str[] ,char origstr[])
{
  /*
    Creates column name.
  */
  strcpy(col[numcols].name,col[lhcol].name ) ;
  strcat(col[numcols].name,str) ;
  strcat(col[numcols].name,col[rhcol].name ) ;

  /*
    Converts column name into numerical equivalent.
  */
  lvalue = lhcol ;
  ltoa(lvalue,nextcol,10) ;
  strcpy(lastcol,nextcol) ;
  strcat(lastcol,str) ;
  lvalue = rhcol ;
  ltoa(lvalue,nextcol,10) ;
  strcat(lastcol,nextcol) ;
  /*
    Replaces all occurances of lastcol in numstring and origstr
    with the numerical value of nextcol.
  */
  lvalue = numcols ;
  ltoa(lvalue,nextcol.10) ;
  replstr(numstrng,lastcol,nextcol) ;
  replstr(origstr,lastcol,nextcol) ;
  numcols ++ ;
}
```

```
/********************************************************************

 FUNCTION :      error_response
 CALLED BY:      display_loop
 CALLS   :       wcenters
                 display_loop
 MODIFIED :      4/12/90
 PERSON  :       Keith Calcote & Rick Howard
 PURPOSE :       Displays the appropriate error message for known errors

*********************************************************************/

static void error_response(int num)
{
   switch (num){
   case INPUT_ERROR:
      wcenters(10,BLUE|_RED,"Invalid Input");
      break;
   case PAREN_MISMATCH:
      wcenters(10,BLUE|_RED,"Unmatched Paren");
      break;
   case INVALID_CHAR:
      wcenters(10,BLUE|_RED,"Incorrect Character");
      break;
   default:
      break;
   }
   display_loop();
}
```

```
/**********************************************************************

 FUNCTION :      pause
 CALLED BY:      display_loop
 CALLS    :      wcenters
 MODIFIED :      4/12/90
 PERSON   :      Keith Calcote & Rick Howard
 PURPOSE  :      Makes the user press any key to continue in the program

**********************************************************************/

static void pause(void)
{
   char key;  /*  The user's response   */

   wcenters(18,BLINK|YELLOW|_BROWN,"Push any key to continue") ;
   key = getch() ;
}

/**********************************************************************

 FUNCTION :      quit
 CALLED BY:      table
 CALLS    :      exit
 MODIFIED :      4/12/90
 PERSON   :      Keith Calcote & Rick Howard
 PURPOSE  :      Teminates the program

**********************************************************************/

static void quit(void)
{
   exit(0);
}
```

```
/********************************************************************

 FUNCTION :       pre_hlep
 CALLED BY:       table
 CALLS    :       wshadow
 MODIFIED :       4/12/90
 PERSON   :       Keith Calcote & Rick Howard
 PURPOSE  :       Displays a shadow behind the current window

********************************************************************/

static void pre_help(void)
{
  wshadow(LGREY|_BLACK);
  setonkey(0x2d00,quit,0);
}
```

APPENDIX Q

THE CODE: FILE "GLOBAL.H"

```
/*********************************************************************
                    The Discrete Math Tutor (DMT)
              Thesis Project at the Naval Postgraduate School
                1989-1990 by Keith Calcote and Rick Howard

 LIBRARY CALLS:
            NONE

 PROGRAM CALLS:
            NONE

 GLOBAL FUNCTIONS:

 COMPLETED:     4/12/90

 PERSONS:       Rick Howard

 PURPOSE:       Global variables for the Discrete Math Turtor

*********************************************************************/

/* Constants */

#define SHORT_DELAY 11
#define LEN 50
#define PAGEL 1000
#define TRUE 1
#define FALSE 0
#define ESC 0x011B
/*----------------------------------------------------------------*/
```

```
/* miscellaneous global variables */

static int crow,ccol;               /* Indicates the row and column of the cursor  */

static WINDOW w[10];                /* Handles used to identify some windows       */

static int from_lsn = FALSE;        /* Indicates functions called from the lsn.exe
                     program                           */

FILE *current_notebook;             /* Pointer to the user's notebook file       */

static char notebook_name[12];      /* The user's notebook name                  */

int def_number;                     /* Identifies the array element chosen in the
                                       definitions table                   */

int start_up = 1;                   /* Indicates that no lesson has begun yet      */
/*-----------------------------------------------------------------*/

/*definitions table */

static char *definitions[]= {
   "Graph", "Definition 2", "Definition 3", "Definition 4",
   "Definition 5", "Definition 6", "Definition 7", "Definition 8",
   "Definition 9", "Definition 10", "Definition 11", "Definition 12", NULL
};
```

APPENDIX R

THE CODE: FILE "VIDEO.H"

```
/*************************************************************************
                    The Discrete Math Tutor (DMT)
              Thesis Project at the Naval Postgraduate School
                1989-1990 by Keith Calcote and Rick Howard

 LIBRARY CALLS:
            biosequip          Turbo C Lib
            setvparam          CXL Lib
            videoinit          CXL Lib

 PROGRAM CALLS:
            NONE

 VIDEO FUNCTIONS:
            set_video

 COMPLETED:     4/12/90

 PERSONS:       Rick Howard

 PURPOSE:       Sets the correct parameters for any type of monitor

*************************************************************************/

static void set_video(void)
{
  unsigned int eq. data;

  videoinit();
  eq = biosequip();

  data = (eq >> 4) & 3; /* bits 4 & 5 */
```

```
    switch (data)
    {
    case 1:
        setvparam(VP_CGA);
        break; /* 40 column color */
    case 2:
        setvparam(VP_CGA);
        break; /* 80 column color */
    case 3:
        setvparam(VP_MONO);
        break; /* 80 column monochrome */
    }
}
```

APPENDIX S

THE CODE: FILE "FLASH.C"

```
/*********************************************************************
                 The Discrete Math Tutor (DMT)
          Thesis Project at the Naval Postgraduate School
            1989-1990 by Keith Calcote and Rick Howard

 LIBRARY CALLS:
               cleardevice        Turbo C Lib
               clearviewport      Turbo C Lib
               delay              Turbo C Lib
               detectgraph        Turbo C Lib
               getch              Turbo C Lib
               getmaxx            Turbo C Lib
               getmaxy            Turbo C Lib
               grapherrormsg      Turbo C Lib
               graphresult        Turbo C Lib
               initgraph          Turbo C Lib
               itoa               Turbo C Lib
               outtext            Turbo C Lib
               printf             Turbo C Lib
               puts               Turbo C Lib
               randomize          Turbo C Lib
               rectangle          Turbo C Lib
               setkbcolor         Turbo C Lib
               settextstyle       Turbo C Lib
               setviewport        Turbo C Lib
               strcat             Turbo C Lib
               strcpy             Turbo C Lib
 PROGRAM CALLS:
                      NONE

 FLASH FUNCTIONS:
                      NONE

 COMPLETED:           4/12/90

 PERSONS:             Keith Calcote

 PURPOSE:             Provides the user with a set of flash cards that test his
                      knowledge on simple logic statements
************************************************************************/

/* header files */

#include <graphics.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <dos.h>
/*------------------------------------------------------------------*/

/* constants */

#define FONT 0
#define CHSIZE 5
/*------------------------------------------------------------------*/
```

```
/************************************************************************

 FUNCTION :       main
 CALLED BY:       NONE
 CALLS    :       See Declarations
 MODIFIED :       4/12/90
 PERSON   :       Keith Calcote
 PURPOSE  :       See Declarations

***********************************************************************/

main()
{
   int driver ,mode ;                   /* graphics driver & mode number     */

   int lhvalue,rhvalue,operator ;       /* Holds the value to the left hand
                                           side and right hand side of the
                                           operator; either zero or one     */

   int maxx,maxy ;                      /* The largest values in the x & y
                                           direction for any type of video
                                           screen                       */

   int left,top,right,bottom ;          /* Holds the pixel location to the
                                           corresponding region on the
                                           screen                       */

   int answer ;                         /* User's response              */
```

```
                                      /* The results for each of the four
                                         different operators          */
int score[4][2] =
   {
   {
     0,0       }
   ,
   {
     0,0       }
   ,
   {
     0,0       }
   ,
   {
     0,0       }
}
;

char lhchar[5], rhchar[5];            /* Either TRUE or FALSE          */

char opchar[5];                       /* OR, AND, IMPLIES or IFF        */

char anchar[5];                       /*  User's response: Either TRUE
                                          or FALSE                   */

char outputch[15] ;                   /* Flashcard that is presented to
                                         the screen               */

char ch,string[10] ;                  /* Multi-purpose character array   */


/*
   Initialize the PC graphics drivers.
*/
detectgraph ( &driver , &mode) ;
initgraph ( &driver, &mode , NULL) ;
```

```
/*
   Check for errors on graphics driver initialization.
*/
error = graphresult();
 if (error){
    printf("Error %d \n", error);
    errptr = grapherrormsg(error);
    puts(errptr);
    exit(1);
 }

 /*
    Set the background screen color to BLUE.
 */
 setbkcolor( 1 ):

 /*
    Retrieve the largest x & y values for the user's screen.
 */
 maxx = getmaxx() ;
 maxy = getmaxy() ;

 /*
    Sets the coordinates for the flashcard.
 */
 left = maxx/6 :
 right = maxx* 5/6 ;
 top = maxy /3 ;
 bottom = maxy * 2/3;
 randomize() ;

 /*
    Set the font.
 */
 settextstyle(FONT+1,0,CHSIZE) ;
```

```
/*
  Draw the opening screen.
*/
moveto(maxx/2,maxy/2-75) ;
settextjustify(1,1) ;
outtext("Welcome to ") ;
moveto(maxx/2,maxy/2+20) ;
settextstyle(FONT+1,0,4) ;
outtext(" Basic Truth Table Practice ") ;
moveto(maxx/2,maxy-20) ;
settextstyle(FONT,0,1) ;
outtext("Q (quit), Any other key begins") ;

/*
  Get the user's response.
*/
ch = getch() ;
if(ch != 'q' && ch !='Q')
{
  while(ch != 'q'&& ch !='Q')
  {
    cleardevice() ;
    rectangle(left, top, right, bottom) ;

    /*
      Randomly choose the value of the
      left side of the equation.
    */
    lhvalue = random(2) ;
    switch(lhvalue)
    {
      case 0 :
      strcpy(lhchar, " F ") ;
      break ;
    default :
      strcpy(lhchar, " T ") ;
    }
```

```
/*
   Randomly choose the value of the
   right side of the equation.
*/
rhvalue = random(2) ;
switch(rhvalue)
{
case 0 :
   strcpy(rhchar," F ") ;
   break ;
default :
   strcpy(rhchar," T ") ;
}

/*
   Randomly choose the equation operator and
   determine the correct answer for the equation.
*/
operator = random(4) ;
switch(operator)
{
case 0 :
   strcpy (opchar,"/\\") ; /* and */
   answer = (lhvalue && rhvalue) ;
   break :
case 1 :
   strcpy (opchar,"\\/") ; /* or */
   answer = (lhvalue || rhvalue) ;
   break ;
case 2 :
   strcpy (opchar,"==>") ; /* if...then */
   if(lhvalue == 0)
      answer = 1 ;
   else if(rhvalue == 0)
      answer = 0 ;
   else
      answer = 1 ;
   break ;
case 3 :
   strcpy (opchar,"<==>") ; /* iff */
   if(lhvalue == rhvalue)
      answer = 1 ;
   else
      answer = 0 ;
   break ;
}

/*
   Prepare the answer for display.
*/
switch(answer)
{
case 0 :
   strcpy(anchar," F ") ;
   break ;
default :
   strcpy(anchar," T ") ;
}

/*
   Display the equation.
*/
strcpy(outputch,lhchar) ;
strcat(outputch,opchar) ;
strcat(outputch,rhchar) ;
settextstyle(FONT,0,CHSIZE) ;
moveto(maxx/2,maxy/2) ;
settextjustify(1,1) ;
outtext(outputch) ;

/*
   Get the user's answer to the equation.
*/
settextstyle(FONT, 0, 1) ;
moveto(left, bottom+top/4) ;
settextjustify(0,0) ;
outtext("press: T (true), F (false), Q (quit)") ;
moveto(left, bottom+top/2) ;
setviewport(left,bottom+top/2,left+50,bottom+top/2+25,0) ;
ch = getch() ;
```

```
/*
   Based upon the user's answer, inform him if he was
   correct or incorrect.
*/
switch(ch)
{
case 'T' :
case 't' :
   if(answer == 1)
   {
      score[operator][0] ++ ;
      outtext(" CORRECT ") ;
      delay(2000) ;
      clearviewport() ;
   }
   else
      {
      score[operator][1] ++ ;
      outtext(" WRONG ") ;
      delay(2000) ;
      clearviewport() ;
   }
   break ;
case 'F' :
case 'f' :
   if(answer == 0)
   {
      score[operator]|0] ++ ;
      outtext(" CORRECT ") ;
      delay(2000) ;
      clearviewport() ;
   }
   else
      {
      score[operator][1] ++ ;
      outtext(" WRONG ") ;
      delay(2000) ;
      clearviewport() ;
   }
   break ;
```

```
  case 'q' :
  case 'Q' :
    ch = 'q' ;
    break ;
  default :
    setviewport(0,0,maxx,maxy,0) ;
    moveto(left, bottom+top/2) ;
    outtext("incorrect entry QUITTING" ) ;
    ch = 'q' ;
  }
  setviewport(0,0,maxx,maxy,0) ;
}
```

```
    /*
      When the user quits, display his results.
    */
    cleardevice() ;
    moveto(0,20) ;
    outtext("AND  ") ;
    itoa(score[0][0],string,10) ;
    outtext(string) ;
    outtext(" correct  ") ;
    itoa(score[0][1],string,10) ;
    outtext(string);
    outtext(" incorrect  ") ;
    moveto(0,40) ;
    outtext("OR   ") ;
    itoa(score[1][0],string,10) ;
    outtext(string) ;
    outtext(" correct  ") ;
    itoa(score[1][1],string,10) ;
    outtext(string);
    outtext(" incorrect  ") ;
    moveto(0,60) ;
    outtext("IMPLY ") ;
    itoa(score[2][0],string,10) ;
    outtext(string) ;
    outtext(" correct  ") ;
    itoa(score[2][1],string,10) ;
    outtext(string);
    outtext(" incorrect  ") ;
    moveto(0,80) ;
    outtext("IFF  ") ;
    itoa(score[3][0],string,10) ;
    outtext(string) ;
    outtext(" correct  ") ;
    itoa(score[3][1],string,10) ;
    outtext(string);
    outtext(" incorrect  ") ;
    getch() ;
   }
  closegraph() ;
}
```

APPENDIX T

THE CODE: FILE "HELP.H"

```
/*********************************************************************
               The Discrete Math Tutor (DMT)
          Thesis Project at the Naval Postgraduate School
            1989-1990 by Keith Calcote and Rick Howard

 LIBRARY CALLS:
           NONE

 PROGRAM CALLS:
           NONE

 HELP FUNCTIONS:
          NONE

 COMPLETED:     4/12/90

 PERSONS:       Rick Howard

 PURPOSE:       Identifies constants that relate directly to each help screen
                in the Discrete Math Tutor

*********************************************************************/

/* help category numbers */

#define H_INITIAL                    1
#define H_USER_INTERFACE             2
#define H_START_LSN                  3
#define H_RETURN_TO_LAST_LSN         4
#define H_SSN                        5
#define H_LOGIC                      6
#define H_LSN_HELP                   7
#define H_DEFINITIONS                8
#define H_EXAMPLES                   9
#define H_THEOREMS                   10
#define H_SELECT                     11
#define H_PICTURES                   12
```

```
#define H_REFERENCE                       13
#define H_CALCULATOR                      14
#define H_PROBLEM_SOLVER                  15
#define H_VENN_DIAGRAM_PICS               16
#define H_TRUTH_TABLE_REF                 17
#define H_CALCULATOR_HELP                 18
#define H_TRUTH_TABLE_PROBLEM_SOLVER 19
#define H_TRUTH_TABLE_DRILL               20
#define H_TRUTH_TABLE_RULES               21
#define H_VIEW_NOTEBOOK                   22
#define H_VIEW_NOTEBOOK_HELP              23
#define H_PRINT_NOTEBOOK                  24
#define H_SAVE_POSITION                   25
#define H_EXIT                            26
#define H_UNAVAILABLE                     27
#define H_EXAMS                           28
```

LIST OF REFERENCES

1. "Irrisistable VGA", *Byte*, v. 15, n. 3, March 1990.
2. "Mac IIfx", *Byte*, v. 15, n. 4, April 1990.
3. "Mainstream Amiga", *Byte*, v. 15, n. 5, May 1990.
4. "Memory Management", *Dr. Dobb's*, v. 15, n.5, May 1990.
5. Smedley, M., *CXL: The C Programmer's Extended Function Library*, v. 5.1, pp.1-157, Mike Smedely, 1989.
6. Constantine, L. and Yourdan, E., *Structures Design*, Prentice Inc., 1978.
7. Dede, C., "A Review and Synthesis of Recent Research in Intelligent Computer-Assisted Instruction," *International Journal of Man-Machine Studies*, v. 24, pp. 329-353, 1986.
8. Duchastel, P., "Designing Intelligent Learning Environments," *Aspects of Educational Technology*, v. 21, pp. 93-98.
9. Duchastel, P., "ICAI Systems: Issues in Computer Tutoring," *Computers in Education*, v. 13, pp. 95-100, 1989.
10. Duchastel, P., "Research Directions for ICAI in Canada," *Intelligence Artificielle au Canada*, pp. 16-19, July 1988.
11. Duchastel, P., and Imbeau, J., "Intelligent Computer-assisted Instruction (ICAI): Flexible Learning through Better Student-Computer Interaction," *Journal of Intelligent Tutoring*, v.2, pp. 102-105, 1988.
12. Gane, C., "Data Design in Structured Systems Analysis,"*Info Tech State of the Art Report on Data Design"*, 1980.
13. Gane, C. and Sarson, T., *Structured Systems Analysis: Tools and Techniques*, Prentice Hall Inc., 1978.
14. Ford, L., "Teaching Strategies and Tactics in Intelligent Computer aided Instruction," *Artificial Intelligence Review*, v. 1, pp. 201-215, 1987.
15. Good. R., "Artificial Intelligence and Science Education," *Journal of Research in Science Teaching*, v. 24, pp. 325-342, April 1987.
16. Hansen,a., *C Programming, p. 336, Addison-Westly, 1989.*
17. Jones, M., "Applications of Artificial Intelligence within Education," *Computers and Mathematics with Applications*, v. 11, pp. 517-526, May 1985.

18. Kelly,S.B., *Mas:ering WordPerfect 5,* SYBEX Inc., 1988.

19. Multimate International Corporation, *Multimate Advanced User's Manual,* Multimate International Corporation, 1984.

20. Ok-choon, P., and Seidel, R. J., "Instructional Design Principles and AI Techniques for Development of ICAI," *Computers in Human Behaviour*, v. 3, pp. 273-287, 1987.

21. Seidel, R. J., Ok-choon, P., and Perez, R S., "Expertise of ICAI: Development Requirements," *Computers ini Human Behaviour*, v. 4, pp. 235-256, 1988.

22. Sleeman, D., and Brown, J. S., *Intelligent Tutoring Systems,* Academic Press, 1982.

23. Wenger, *A.I. & Tutoring Systems*, pp 12-45, 1987.

24. Winograd, T., and Fernando, F., *Understanding Computers and Cognition*, Ablex Publishing Corporation, 1986.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
Cameron Station
Alexandria, VA 22304-6145

2. Library, Code 0142 2
Naval Postgraduate School
Monterey, CA 93943-5002

3. Department Chairman, Code MA 1
Department of Mathematics
Naval Postgraduate School
Monterey, CA 93943-5100

4. Department Chairman, Code CS/Mz 1
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943-5100

5. Professor Man-tak Shing, Code CS/Sh 2
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943-5100

6. Professor Kim Hefner, Code MA/Hk 2
Department of Mathematics
Naval Postgraduate School
Monterey, CA 93943-5100

7. Capt. Richard A. Howard 2
15 Villa Place
Eatontown, NJ 07724

8. Lieutenant Roy K. Calcote 2
c/o Mary Jane Bales
PO Box 637
Devine, TX 78016